J R C T E C H N I C A L R E P O R T S

The Alpha-Tree Algorithm

Theory, Algorithms, and Applications

Georgios K. Ouzounis and Pierre Soille

2012

Report EUR 25500 EN

Joint
Research
Centre

European Commission
Joint Research Centre
Institute for the Protection and Security of the Citizen

Contact information:
Pierre Soille
Address: Joint Research Centre, Via Enrico Fermi 2749, TP 267, 21027 Ispra (VA), Italy
E-mail: Pierre.Soille@jrc.ec.europa.eu
Tel.: +39 0332 78 5068
Fax: +39 0332 78 5154

http://ipsc.jrc.ec.europa.eu/
http://www.jrc.ec.europa.eu/

tre of the European Commission.

JRC74511

EUR 25500 EN

ISBN 978-92-79-26279-1

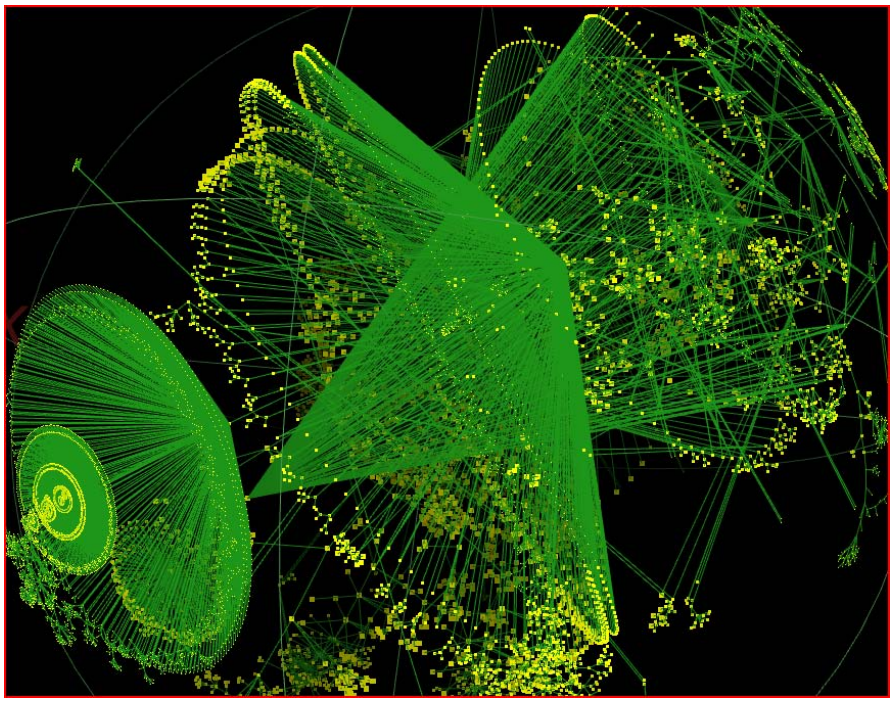
ISSN 1831-9424

doi:10.2788/48773

Luxembourg: Publications Office of the European Union, 2012

The Alpha-Tree Algorithm

– Theory, Algorithms, and Applications –



Georgios K. Ouzounis and Pierre Soille

European Commission, Joint Research Centre

Contents

Abstract

A new multi-scale graph-space connectivity framework is presented. It is based upon a measure of dissimilarity between adjacent elements of the graph that is used to construct a hierarchy of partitions. Connected components or partition cells are either preserved or rejected based on a set of attribute criteria that are enumerated through logical predicates. Enforcing attribute constraints generates a dichotomy of the partition hierarchy that can be used for image segmentation and other related applications. The framework is supported by an efficient union-find based algorithm that delivers a tree representation of the totally ordered set of graph-space partitions. It is referred to as the Alpha-Tree. The practical complexity of the algorithm is linear with respect to the number of pixels. Processes on the tree can be launched interactively and in real-time, from a separate module detached from the tree construction phase. The type of attributes and attribute thresholds can be set and adjusted interactively. Timed experiments on highly complicated and massive satellite image tiles are presented, complemented by comparisons to the standard method.

Chapter 1
Introduction

Image segmentation is a transformation often described as the partitioning of the image domain into a set of meaningful regions according to some pre-specified criteria. Methods for computing this exist by large in literature and may vary significantly between them [PP93]. Among the plethora of the different approaches, hierarchical methods deserves special attention because they yield a family of fine to coarse partitions rather than a single output. Hierarchical methods are often considered richer in descriptive ability and more likely to contain meaningful and spatially consistent regions than methods delivering a single partition or a family of unordered partitions. Any hierarchy can be represented as a tree data structure that is commonly referred to as a *dendrogram* in both classification and cluster analysis. In the case of hierarchical segmentation, all leaf nodes occur at the same level, called the base level of the hierarchy, and they match the segments of the finest partition delivered by the considered method. Subsequent levels of the hierarchy occur when at least two adjacent segments merge to form a larger segment. However, not all tree-based image representations can be used for the purpose of hierarchical segmentation. An example is the component tree [Jon97; Jon99] (also called max-tree [SOG98]), whose node ordering matches the nesting of image peak components. Each node corresponds to sets of at zones [SS95] for which there exists a unique mapping to a peak component. A leaf node in this representation corresponds to a regional maximum, i.e. a peak component that defines a single at zone of the same extent. Evidently, the component tree represents the hierarchy of level sets rather than a hierarchy of partitions. Component trees are widely used for computing attribute filters [BJ96; URW07], pattern spectra [URW07; OW10], multi-scale decompositions [OS10; OPS12], interactive segmentation [Pas+11], etc.

Horowitz and Pavlidis [HP76] were among the first to propose a hierarchical image segmentation technique using a split and merge algorithm. Numerous other methods have been developed since, that proceed typically by first defining either the finest (base) or the coarsest (top) level of the hierarchy. A merging or splitting order is then defined to generate the subsequent levels of the hierarchy. Most modern methods proceed from fine to coarse levels and the underlying merging order relies on a dissimilarity measure computed for all pairs of adjacent segments. In some

cases a region model is necessary to compute the dissimilarity between segments that are formed by the union of two or more segments. Two commonly encountered base levels are the image definition domain partitioned in its entirety into a set of singleton sets, or into a set of iso-intensity connected components, i.e. at zones. The latter are sets of path-wise connected elements of maximal extent and of constant intensity. Intensity in the case of scalar images, is the most obvious metric based on which a pixel dissimilarity measure can be defined. Considering the absolute intensity difference, a sequence of progressive merges between adjacent segments, with increasing dissimilarity among them, can be iterated until some stability criterion is reached. The process converges by default when a single segment, covering the whole definition domain, is obtained.

An early example utilising this approach, was implemented for the *hierarchical stepwise optimisation* (HSWO) method proposed in [BG89]. It relies on attributing each individual segment with its mean pixel intensity. This metric is recomputed for each newly created segment and it is used to update the dissimilarity between itself and all its adjacent segments. A similar approach is considered for the binary partition tree, proposed in [SG00; VMS08], in which each new segment model obtained from a pair of two subsets, equals to the largest one in that pair. Binary partition trees however are order dependent and each level of the representation is a strict dichotomy of its predecessor. An order independent best merging extension solving the problems of ties of iterative merging of region pairs is proposed in [Sch95]. Another approach that is also order independent but does not require a region model, is obtained by applying the *single-linkage hierarchical clustering* method [GR69] on the image data. With this approach, first proposed in [NMI79], the dissimilarity between any two segments appearing in the hierarchy is defined as the smallest dissimilarity of the pairs of adjacent pixels, such that the first pixel of each pair belongs to one of the two segments, and the second to the other. The resulting segments are called *quasi-flat zones* [MM00] or λ*-flat zones* [NAJ07] in mathematical morphology. Both terms however are somewhat misleading since the " atness condition" set by the threshold value λ acts only on paths. This often leads to connected components containing adjacent pixels with a dissimilarity larger than the value of λ [Soi11], due to the lack of some control mechanism. To describe this more accurately, Soille [Soi08] introduced the term α*-connected component*, which is essentially a path-wise connected component in which any two adjacent elements along a path differ by no more than a value α.

Hierarchical segmentation in the context of graph theory has been approached with a model in which, the tip of the hierarchy is usually represented by a minimum spanning tree of the image pixels while the successive levels correspond to minimum spanning forests of the corresponding minimum spanning tree. This approach was first proposed in [MLC86] and independently in [Mey94], where it is shown that the watershed of the gradient of an image is equivalent to the minimum spanning forest of the neighbourhood graph of the initial image. Any hierarchy on the neighbourhood graph of an image is called an *ultrametric* watershed in [Naj09]. Hierarchies of region adjacency graphs are also studied in [Nac95] and in [HK03].

In this report, the concept of hierarchical segmentation is approached through the framework of constrained connectivity [Soi08]. Constrained connectivity has several desirable properties among which, is the hierarchical ordering of consecutive partitions subject to a dissimilarity metric and constraining criteria, order independence, and prevention of leakage through transitions. Creating a segmentation of the input was so far dealt with by constraining the evolution of α-connected components [Soi08]. By contrast to this, a rather different strategy is proposed in which the evolution of α-connected components is computed under the absence of constraints and is represented by a tree structure, referred to as the α-tree. Constraints are put in place following the tree creation, through binary attribute criteria, enumerated using logical predicates [Soi07]. The latter can be configured based on a number of different dissimilarity measures. Creating the desired segmentation becomes a similar process to regular attribute filtering, as dealt with in [SOG98; MW02], and the output can be readjusted interactively and in real-time.

This report is structured as follows. In Chap. 2 an overview of the definitions of image connectivity and partitions is given. Chapter 3 introduces a new framework for the concept of dissimilarity-based connectivity, accounting for recent advances in the field and including a detailed presentation of the notions of α-tree and attribute-constrained connectivity. It is followed in Chap. 4 by a detailed description of an efficient algorithm for computing the α-tree structure and operators enforcing constraints. The proposed α-tree algorithm relies on Tarjan s union-find method [Tar75] and employs path-compression [Tar79]. A set of experiments carried out to evaluate the algorithm s performance is detailed in Chap. 5. Indicative applications demonstrating the power of the proposed α-tree representation and its associated algorithm for information extraction on very high resolution satellite images are given in Chap. 6. An overview of the algorithm features and its performance is given in Chap. 7 along with a summary of conclusions.

Chapter 2
Preliminaries on Image Connectivity and Partitions

Let I be a grey-tone image and E be its definition domain, i.e. a Euclidean subspace. A partition $\mathbf{P}$ of E is its division into a set of non overlapping and non-empty *cells*, the union of which is equal to E. The cells of $\mathbf{P}$ are both collectively exhaustive and mutually exclusive with respect to the set being partitioned. The formal definition as given in [Ser88, Chap. 1], is the following:

Definition 1 *Let E be the definition domain of an image. A partition $\mathbf{P}$ of E is a mapping $x \to \mathbf{P}(x)$ from E into the power set of E, denoted by $\mathscr{P}(E)$, such that:*

1. $\forall x \in E \Rightarrow x \in \mathbf{P}(x)$;
2. $\forall x, y \in E \Rightarrow \mathbf{P}(x) = \mathbf{P}(y)$ *or* $\mathbf{P}(x) \cap \mathbf{P}(y) = \emptyset$.

The term $\mathbf{P}(x)$ above indicates a cell of $\mathbf{P}$ marked by/containing a point $x \in E$. Each cell is a cluster of elements of E that are equivalent among themselves according to a certain condition, and given a cell none of its elements is equivalent to any other element from a different cell. It follows that $\bigcup_{x \in E} \mathbf{P}(x) = E$.

Image partitions can be generated interactively or automatically. An elegant partitioning scheme frequently encountered in image analysis is the separation of the image content into foreground and background components. This dichotomy can be realised considering the connectivity relations among the elements of E, i.e. by defining a partition cell for each connected set of maximal extent. Image connectivity is modelled though *connectivity classes*. The definition given in [Ser88, Chap. 2] states:

Definition 2 *Let E be an arbitrary non-empty space. A connectivity class or connection $\mathscr{C}$ is any family in $\mathscr{P}(E)$ such that:*

1. $\emptyset \in \mathscr{C}$;
2. $\forall\, x \in E, \{x\} \in \mathscr{C}$;
3. *for each family $\{C_i, i \in L\} \subseteq \mathscr{C}$, $\bigcap_i C_i \neq \emptyset$ implies $\bigcup_i C_i \in \mathscr{C}$, where L is an index set.*

A connected set $C \subseteq E$ is called a *connected component* if there is no other connected set $C' \supset C$ such that $C' \subseteq E$ and $C' \in \mathscr{C}$. Connected components can

be extracted by connected operators. The latter form the basis of attribute filters [BJ96] which are edge preserving operators preserving connected components that satisfy some attribute criterion. Connected operators can be customised to address more general notions of connectivity such as clustering, contraction [Ron98; BG02], mask-based connectivity [OW07] or partition-induced connectivity [Ser06; OW10; WO10].

Chapter 3
α-Tree Representation and Attribute-Constrained Connectivity

Dissimilarity metrics between pixels of an image projected in a graph space are recalled in Sec. 3.1. These metrics are at the basis of the notion of α-connectivity detailed in Sec. 3.2. The underlying α-tree representation is presented in Sec. 3.3. Finally, the notion of attribute-constrained connectivity is proposed in Sec. 3.4.

3.1 Dissimilarity metrics

Assume that an image I is projected on a graph space in which vertices correspond to pixels, and edges to pairs of adjacent vertices. Each element x of E can be addressed as a singleton set $\{x\}$ that is characterised by some attribute $Attr(x)$ such as its position, intensity, etc. Dissimilarity measures d, computed between attributes of adjacent elements are used for clustering or segmentation, subject to some thresholding criterion. A common dissimilarity measure between two elements is the L_p norm of the difference of the attribute vectors associated with these elements. Typical choices for p are 1,2 or ∞. Another common choice for multispectral images is the spectral angular distance [Kru+93]. A comprehensive presentation of dissimilarity measures employed in pattern recognition is given in [PD05]. Other dissimilarities are developed in [Soi11] to prevent chaining along transitions (ramps) while favouring it along homogeneous regions, and in [GS11] for taking into account overall image statistics such as the co-occurrence frequency or the local mutual information. The dissimilarity between an element and itself is always null: $d(x,x) = 0$ for all $x \in E$.

In the case that x and y are not adjacent, their dissimilarity can be determined by computing the Euclidean or the path-wise distance between the respective element attributes. A path $\pi(x \rightsquigarrow y)$ between any two elements $x, y \in E$ is a chain of pairwise adjacent elements:

$$\pi(x \rightsquigarrow y) \equiv \langle x = x_0, x_1, ..., x_{N-1} = y \rangle, \tag{3.1}$$

in which, N is the number of elements in the path.

De nition 3 *Let $\Pi \neq \emptyset$ be the set of all possible paths between the two elements x and y. The minimum dissimilarity metric with respect to some pre-specified element attribute, is the ultrametric functional $d^{\wedge}$ given by:*

$$d^{\wedge}(x,y) = \bigwedge_{\pi \in \Pi} \Big\{ \bigvee_{i \in [0,\ldots,N_\pi - 1]} \Big\{ d(x_i, x_{i+1}) \mid x_i, x_{i+1} \in \pi \Big\} \Big\}. \tag{3.2}$$

In words, (3.2) states that the dissimilarity measure between any two path connected elements of E is the infimum among the set of values, each corresponding to the maximal dissimilarity between pairwise adjacent elements along each path.

3.2 α-Connectivity Revisited

The minimum dissimilarity metric of Def. 3 has been used to define single-linkage [NMI79] and α-connected components [Soi08], that are also known as *quasi-flat zones* [MM00]. An α-connected component α-CC(x) marked by/containing a point $x \in E$ is defined as:

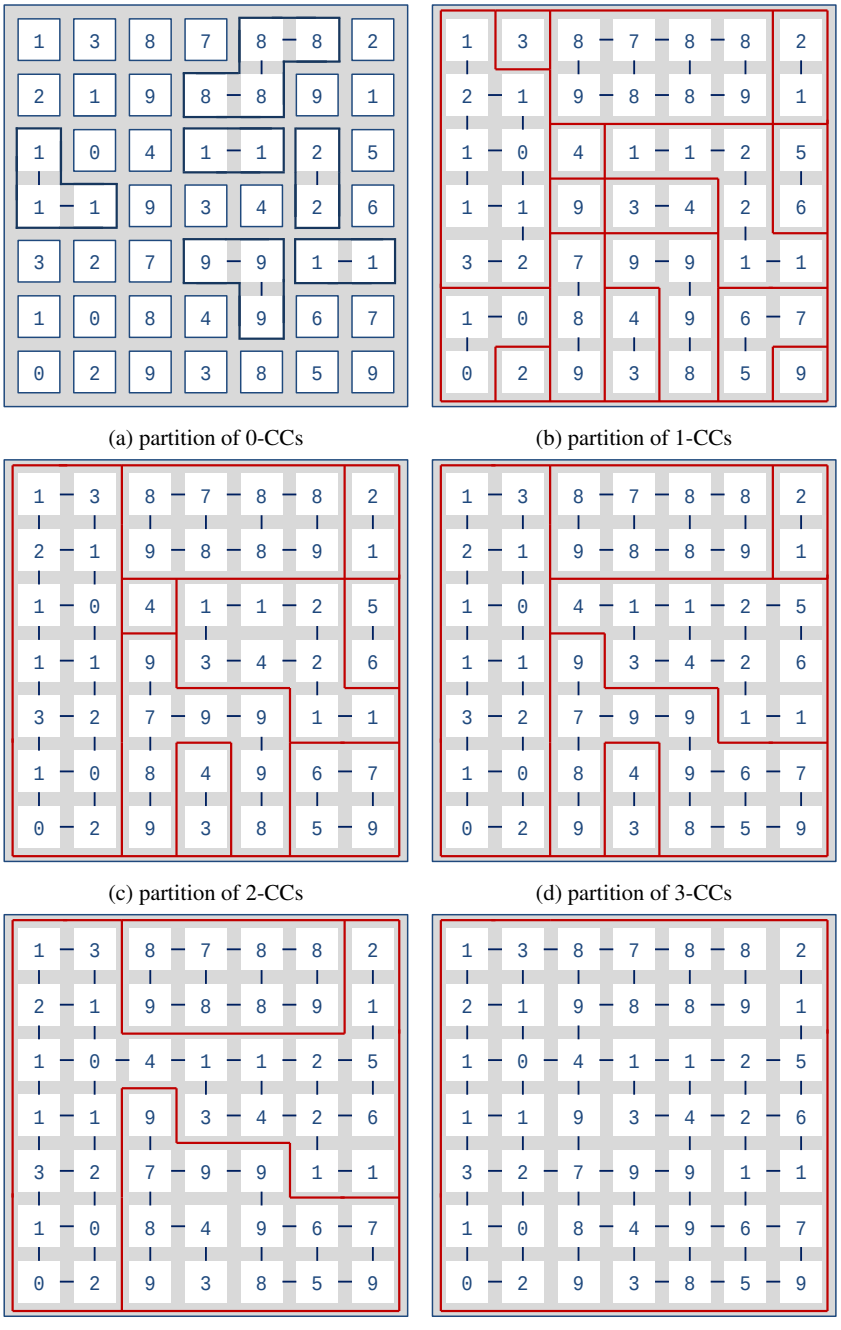
$$\begin{aligned} \alpha\text{-CC}(x) = \{x\} \cup \{y \mid \exists\, \pi(x \rightsquigarrow y) :\ \forall\, x_i \in \pi(x \rightsquigarrow y) \wedge \\ x_i \neq y, d(x_i, x_{i+1}) \leq \alpha\}, \end{aligned} \tag{3.3}$$

or, equivalently as:

$$\alpha\text{-CC}(x) = \{y \mid d^{\wedge}(x,y) \leq \alpha\}. \tag{3.4}$$

Equation (3.3) states that all pixels are connected sets themselves, i.e. they are α-connected to themselves, and a connected set of maximal extent marked by a point $x \in E$ is the union of x with all points $y \in E$ such that for each one there exists a path from x to y in which all adjacent elements have a dissimilarity less than or equal to α. Note that (3.3) complies with the conditions concerning the subsets of E as given in the definition of connectivity classes, i.e. Def. 2. Complemented by a further condition on the connectivity of the empty set representing the background, α-connectivity forms a sub-connection of the canonical path-wise connection on a graph space, for which the α parameter is a connectivity threshold. If the dissimilarity between any two adjacent elements x and y is less than or equal to α, the two are directly connected, i.e. there exists an edge between x and y, thus are members of the same α-CC. The case in which $d(x,y) > \alpha$ does not imply that x and y do not belong to the same α-CC but only that there is no direct linkage between them. Examples are shown in Fig. 3.1 and in [OS11].

Connected components that consist exclusively of elements for any two of which $d^{\wedge}(x,y) = 0$, are called *reference connected components* or 0-CCs. The case in which the element attribute is the intensity and the dissimilarity measure d is the L_p norm, a reference component marked by x is equivalent to the respective image at zone [SS95] containing x, i.e. a maximal connected iso-intensity image region.



(a) partition of 0-CCs

(b) partition of 1-CCs

(c) partition of 2-CCs

(d) partition of 3-CCs

(e) partition of 4-CCs

(f) partition of 5-CCs

Fig. 3.1 Example of successive α-connected components with $\alpha \in [0,1,2,3,4,5]$, using the absolute intensity difference between adjacent pixels as the dissimilarity measure.

The α-CCs are equivalence classes [Soi08] on the image definition domain, consequently the set of α-CC(x) for all $x \in E$ defines a partition of E, i.e. α-CCs are both collectively exhaustive and mutually exclusive in E. Moreover:

$$\bigcup_{x \in E} \alpha\text{-CC}(x) = E. \tag{3.5}$$

Evidently, greater values of α result in larger α-CCs, i.e. produce coarser partitions of E. Consider a point $x \in E$ and a range A of α values. All respective α-CCs containing x are ordered with respect to α such that:

$$\alpha_i\text{-CC}(x) \subseteq \alpha_j\text{-CC}(x), \forall\ \alpha_i \leq \alpha_j \text{ and } \alpha_i, \alpha_j \in A. \tag{3.6}$$

Figure 3.1 shows an example of a fine to coarse partition hierarchy for 5 α-levels. More precisely, Fig. 3.1a shows the original image partitioned into 0-CCs, and Figs. 3.1b–f the successive partitions for values of α in $[1,2,...,5]$. This order with respect to α allows for the definition of a structured representation of the stack of nested partitions, that is referred to as the *α-tree*.

3.3 The α-tree Representation

Given an image I, let $\{\mathbf{P}\}^A$ be the set of all α-partitions of its definition domain E. The term A is called the alpha dissimilarity range and is a vector of thresholds $A = [0, 1, ..., \alpha_{\max}]$. Given a point $x \in E$ marking a cell of a partition $\mathbf{P}^{\alpha_j} \in \{\mathbf{P}\}^A$, with $\alpha_j \in A$ and assuming that $|A| > 1$, then for any other $\alpha_i \in A$ such that $\alpha_i < \alpha_j$, (3.6) leads to a more general conclusion:

$$\forall x \in E, \alpha_i\text{-CC}(x) \subseteq \alpha_j\text{-CC}(x) \Rightarrow \mathbf{P}^{\alpha_j} \preccurlyeq \mathbf{P}^{\alpha_i}. \tag{3.7}$$

The symbol $\preccurlyeq$ denotes an order relation with respect to $\alpha \in A$ [Soi08]. The family of ordered partitions of E for the entire α dissimilarity range is defined as follows:

De nition 4 *A partition pyramid of E for $|A| > 1$, is a mapping $\triangle^A : E \to \{\mathbf{E}\}^A$ given by:*

$$\begin{aligned} \triangle^A = &\left\{ \mathbf{P}^{\alpha=0}, \mathbf{P}^{\alpha=1}, ..., \mathbf{P}^{\alpha_{\max}} \right\} \ \mid\ \mathbf{P}^{\alpha'} \preccurlyeq \mathbf{P}^{\alpha}, \\ &\forall\ \alpha' < \alpha \textit{ with } \alpha', \alpha \in A. \end{aligned} \tag{3.8}$$

A pyramid level $\triangle^A_\alpha \in \triangle^A$ is a partition $\mathbf{P}^\alpha$ of E, with $\alpha \in A$. Note that the base of the pyramid corresponds to the finest, and the tip to the coarsest partition of E, i.e. to the set of reference components defined at $a = 0$, and to the single α_{max}-CC associated to the image definition domain, respectively.

Partition pyramids have a notable drawback; some α-CCs may persist in more than one level of $\triangle^A$. This introduces a redundancy overhead and an example is

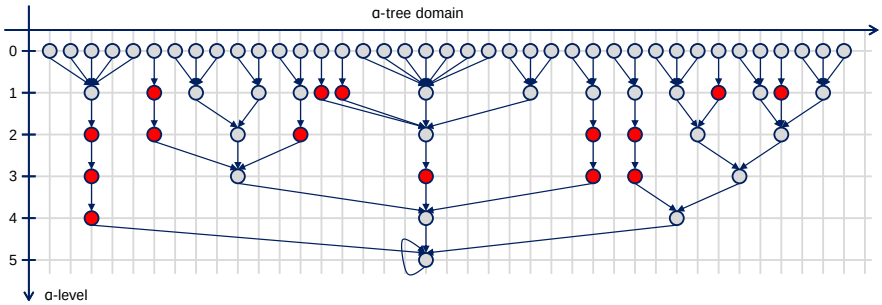


Fig. 3.2 The partition pyramid computed from the stack of partitions of the 7×7 image used in Fig. 3.1. Red nodes illustrate the redundancy contained in the partition pyramid.

given in Fig 3.2, which shows the partition pyramid computed from the top-left image of Fig. 3.1. α-CCs are shown as circles, and those marked with red highlight the redundancy. They are replicas of α-CCs that appeared at smaller α levels and persist unmodified until a merge occurs with some other α-CC at a higher level. To counter this, an index mapping of α-CCs is introduced, that leads to a hierarchical partition representation structure configured with strict inclusion. Consider a variable $j \in J^\alpha$, in which $J^\alpha \subseteq \mathbb{Z}$ is an index set, employed to address the α-CCs making up $\mathbf{P}^\alpha$. Given a point $x \in E$, there exists explicitly only one $j \in J^\alpha : x \in \alpha_j$-CC.

De nition 5 *Let $\triangle^A$ be an α-partition pyramid of a grey-tone image I, defined for a dissimilarity measure range A. An α-partition hierarchy Π^A is a family of ordered mappings $\Pi^A_\alpha : J^\alpha \to K^\alpha$ with $K^\alpha \subseteq J^\alpha$, given by:*

$$\begin{aligned} \Pi^A = \Big\{ \Pi^A_{\alpha=0}, \Pi^A_{\alpha=1}, \ldots, \Pi^A_{\alpha_{\max}} \Big\} \mid \Pi^A_{\alpha'} \prec \Pi^A_\alpha, \\ \forall\, \alpha' < \alpha \text{ with } \alpha', \alpha \in A, \end{aligned} \tag{3.9}$$

and $\forall \alpha \in A \setminus 0$ and $\forall j \in J^\alpha$:

$$\Pi^A_\alpha = \Big\{ \alpha_j\text{-CC} \mid \Big(\alpha_j\text{-CC} \in \triangle^A_\alpha\Big) \wedge \Big(\alpha_j\text{-CC} \notin \triangle^A_{\alpha-1}\Big) \Big\}. \tag{3.10}$$

In words, each level of the hierarchy Π^A contains explicitly only those elements of the corresponding pyramid level, that appear for the first time.

The α-partition hierarchy Π^A is a lossless compression of an α-partition pyramid. Each level $\triangle^A_\alpha$ can be restored as follows:

$$\triangle^A_\alpha = \Big\{ \bigvee_{\alpha' \in [0,\ldots,\alpha],\ j \in J^{\alpha'}} \alpha'_j\text{-CC} \mid \alpha'_j\text{-CC} \in \Pi^A \Big\}, \tag{3.11}$$

i.e. it is the set of all maximal α'-CC : $\alpha' \leq \alpha$, which further define a partition of E.

The α-partition hierarchy is called the α-*tree*, and is a spatially rooted dendrogram, also known as 3D dendrogram [Soi09]. The α-tree of the example in Fig. 3.1 can be obtained from Fig. 3.2 by removing the redundant nodes (i.e., the red nodes) and redirecting children-to-parent edges accordingly. The resulting α-tree is shown in Fig. 3.3.

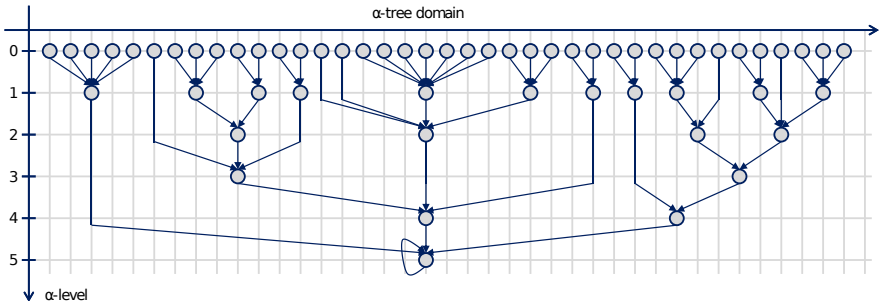


Fig. 3.3 The α-tree (partition hierarchy) of the 7×7 image used in Fig. 3.1. Note that it does not contain any redundant node, contrary to the corresponding partition pyramid shown in Fig. 3.2.

Comparing α-trees against other, conceptually similar hierarchical partition representation structures, like the ultrametric watershed [Naj09; Naj11] and its closely related min-tree (dual of the max-tree structure [SOG98]) of the *edge graph* of an image, also called the *line graph*, a link is observed as detailed in [NS10; SN12]. Consider an image represented in a graph-space from which its edge graph is computed. Moreover, let the weight of each node of the edge graph be the dissimilarity measure between the pair of adjacent pixels it represents. Computing a min-tree from the node weights yields a similar but not identical structure to the α-tree, in which the smallest connected components map to at least two pixels (each node of the edge graph correspond to two pixels of the input image). Consequently at zones, which define the leaves of an α-tree, are not included in this representation. This is demonstrated in Fig. 3.4 on a sample tile of a binary, 4-connected chessboard pattern shown in (a). let the dissimilarity measure be the absolute difference among adjacent pixels. The α-tree of image (a) contains 25 nodes at its base level (the 25 0-CCs of the pattern) and one node for all subsequent levels since all the nodes are merged in a single step, i.e. for a dissimilarity threshold value greater or equal to 1. By contrast, the edge graph of the pattern contains 40 nodes shown in image (b), all assigned the value 1. The min-tree of this pattern contains a single node only since all threshold sets are identical. This can be corrected for by doubling the nodes of the initial image and linking the resulting pairs of identical nodes with edges of null-dissimilarity. In this case the min-tree of the edge graph, computed from the expanded input graph accords with the α-tree representation [SN12]. However, for an image of $n \times m$ pixels like in (a), the number of vertices of the initial graph is

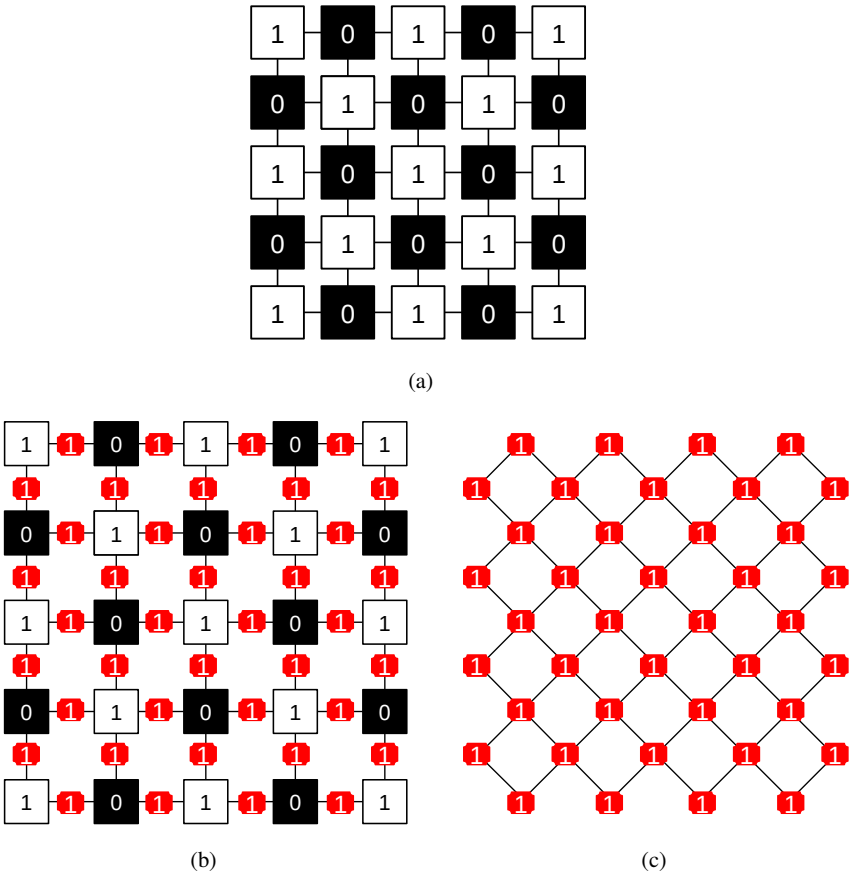


Fig. 3.4 Edge graph of a chessboard pattern. (a) The input image with elements being 4-connected. (b) Any two adjacent element differ exactly by 1 (red labels). (c) The 4-connected edge graph.

$n \times m$ as required for the leaves of the α-tree, by contrast to $3(n \times m) - m - n$ nodes of the resulting edge graph as required for the min-tree after node doubling!

3.4 Attribute-Constrained Connectivity

The definition of connected components as sets of maximal extent yields a maximal partition of the image space, which in certain applications induces a problem often referred to in literature as "leakage" [OW10]. In the case of connectivity relations based on dissimilarity measures, leakage describes the existence of paths between regions that should be treated separately, in which adjacent elements differ less than α. This undesired effect can be countered by constraining the range of

some generic attribute describing the components in question. The introduction of constraints in the context of image segmentation prevents the formation of maximal partitions [OS11] and thus contradicts the framework of connective segmentation criteria discussed in [Ser06]. This works to the benefit of image segmentation since it allows for strict control over the segmentation process. Examples are in [Soi07; Soi08].

Constrained connectivity was originally proposed in [Soi08] as a method to partition the image definition domain into unique maximal connected components satisfying a series of constraints. Since then, it has matured into a more general framework incorporating arbitrary constraints expressed through logical predicates as well as arbitrary dissimilarities, further to the local and global absolute intensity differences initially proposed. Some of these developments were presented in [Soi07; SG08; SG09; Soi11; OS11; GS11], and complemented by recent advances, are unified into a single framework in this section. The concepts presented address elements of some image definition domain but are equally applicable to any graph-space data representation.

Using first-order-logic [Bar77] as our symbolic formal system [And02], logical predicates are defined as Boolean-valued functions that return **true** when the associated argument satisfies the predicate and **false** otherwise. Logical predicates P of α-CCs are typically *decreasing* ($P^{\downarrow}$). If not, they are addressed as *non-decreasing*. A predicate is decreasing if and only if, for any two sets $X, Y \subseteq E$, the following condition holds:

$$P^{\downarrow}(X) = \textbf{true} \Rightarrow P^{\downarrow}(Y) = \textbf{true},\ \forall\, Y \subseteq X. \tag{3.12}$$

Moreover, decreasing logical predicates are imposed to return **true** on any reference component. This ensures that there exists at least one α-CC that satisfies the predicate for every pixel $x \in E$. By relaxing this condition, the resulting connected components form a partial partition [Ron08; Ron11a; Ron11b].

De nition 6 *Let $\{P_n^{\downarrow}\}$ be a set of decreasing predicates, with $n \in N$ and $N \geq 1$, returning **true** on every reference connected component of E. An* attribute-constrained component *on E, containing $x \in E$ is defined as [Soi07] :*

$$\begin{aligned}&(P_1^{\downarrow}, ..., P_N^{\downarrow})\text{-CC}(x) = \\ &\bigvee \Big\{ \alpha_i\text{-CC}(x) \mid\ P_n^{\downarrow}\Big(\alpha_i\text{-CC}(x)\Big) = \textbf{\textit{true}},\ \forall\, n \in N \Big\}.\end{aligned} \tag{3.13}$$

The example of (α, ω)-connectivity based on this definition, employs two decreasing predicates $P_1^{\downarrow}$ and $P_2^{\downarrow}$, the first one of which returns **true** if the maximally indexed component α_i is less than or equal to the dissimilarity threshold α and the second if its total element attribute variation is less than or equal to the global range parameter ω. The predicate $P_2^{\downarrow}$ appears often as a regularisation factor [Soi08; SG08; SG09]. Soille [Soi07] proposed the use of a number of other attributes and demonstrated an example of image simplification with a variance-based predicate. In this work, we formalise the use of any attribute in either decreasing or non-decreasing predicates.

Consider the predicate P applied to the α-CC of any given pixel x defined as:

$$P(\alpha\text{-CC}(x)) = \begin{cases} \textbf{true} & \text{if } Attr(X) \leq \tau \text{ or } \alpha = 0, \\ \textbf{false} & \text{otherwise,} \end{cases} \tag{3.14}$$

with τ being an attribute threshold. Increasing attributes yield decreasing predicates since the condition $Y \subseteq X \Rightarrow Attr(Y) \leq Attr(X)$ complies with (3.12). Consider a logical predicate based on the area metric for example. According to Def. 6, it returns **true** if the number of pixels of the α_i-CC(x) does not exceed a given threshold value τ, and **false** otherwise. Moreover, by default $P^{\text{area}}(0\text{-CC}(x))$=**true** for all x and irrespective of whether $Area(0\text{-CC}(x)) \leq \tau$ or not.

Non-increasing attributes such as shape descriptors, often provide a more delicate characterisation of the structures of interest and find usage in many application domains. Predicates in this case are non-decreasing since (3.12) is not satisfied. Therefore, there is no guarantee that all ancestors of an α-CC satisfying a given predicate also satisfy it. To counter this, an additional condition is required, reasoning the constraint. There can be several such conditions that are referred to as *constraining rules*. Soille in [Soi07; SG09] presented the *Max-Rule* which selects the largest α_i-CC with an attribute measure smaller than or equal to τ *and* all its ancestors satisfy the same predicate.

Definition 7 *Let $\{P_n\}$ be a set of decreasing predicates, with $n \in N$ and $N \geq 1$, returning **true** on every reference connected component of E. An* attribute-constrained component *on E, containing $x \in E$ and configured with the Max-Rule is defined as [Soi07; SG09]:*

$$\begin{aligned} &(P_1, \ldots, P_n)\text{-CC}(x) = \\ &\bigvee \Big\{ \alpha_i\text{-CC}(x) \;\Big|\; P_n\Big(\alpha_i\text{-CC}(x)\Big) = \textbf{\textit{true}}\ \forall n \in \{1, \ldots, N\} \\ &\text{and } P_n\Big(\alpha_k\text{-CC}(y)\Big) = \textbf{\textit{true}}\ \forall k \leq i \text{ and all } y \in \alpha_i\text{-CC}(x)\} \Big\}. \end{aligned} \tag{3.15}$$

Note that P_n returns **true** on every reference connected component of E, irrespective of its attribute measure. An example of non-structural and non-increasing attribute is the intensity variance. In the case of absolute difference dissimilarity, the variance of any 0-CC is obviously 0 while it is strictly greater for any other α-CC with $\alpha > 0$. Examples of other non-increasing attributes are the perimeter, compactness, elongation, rectangularity, entropy, moment invariants, the mean value of the gradient magnitude, etc.

Chapter 4
The α-tree Algorithm

Image segmentation based on constrained connectivity relations was so far dealt with by constraining the evolution of α-CCs. In this chapter we propose a new approach; the evolution of α-CCs is computed under the absence of constraints for the entire range of threshold values α, and is encoded on the α-tree structure. Single or multiple constraints are put in place in a separate stage, through binary attribute criteria on the respective α-CCs. This allows for interactive segmentation, graph filtering, and computation of α-connected pattern spectra for texture analysis [OS11].

An algorithm for computing efficiently the α-tree data structure is presented. It is designed in a modular architecture in which, tree construction, tree processing, and output image restitution constitute a separate module each. Like other similar, state-of-the-art image representations such as the max tree/component tree [SOG98; Jon97; Jon99], once the tree is computed, operations can be launched interactively without the need of recomputing the structure. The α-tree construction relies on Tarjan s union-find algorithm [Tar75; Tar83] both for labelling the 0-CCs during initialisation, and for merging consecutive α-CC during its run-time. It is optimised according to related algorithmic advances in [DST92; FG96] and morphological methods in [WR00; MW02; Hes03; Gér05; NC06; Ber+07].

This chapter is organised as follows. A brief introduction on the union-find algorithm and a presentation of the α-tree data structure are presented in Secs. 4.1 and 4.2 respectively. The α-tree construction with its labelling of reference connected components and wavefront propagation steps are detailed in Sec. 4.3. Once the tree structure is constructed, it can be processed by selecting nodes satisfying a series of constraint while mapping them back onto the image domain. The corresponding tree processing and restitution steps are detailed in Sec. 4.4.

Algorithm 1 Union-Find Procedures

```
Procedure: MakeSet()
Require: var p;
 1: return Parent[p] = -1;
Procedure: FindRoot()
Require: var p, par_p;
 1: par_p=Parent[p];
 2: if par_p ≥ 0 then
 3:     par_p = FindRoot(par_p);
 4:     return par_p;
 5: else
 6:     return p;
 7: end if
Procedure: Criterion()
Require: var p, q, α;
 1: var d = ComputeDiss(p, q);
 2: if d ≤ α then
 3:     return true;
 4: else
 5:     return false;
 6: end if
Procedure: Union()
Require: var p, q, α;
 1: var r = FindRoot(q);
 2: if r != p then
 3:     if Criterion(p, r, α)== true then
 4:         Parent[r] = p;
 5:         return true;
 6:     else
 7:         return false;
 8:     end if
 9: end if
```

4.1 Union-Find Preliminaries

A partition of the image domain is represented by a set of separable and non-overlapping regions, the union of which returns the original image. These regions are referred to as disjoint sets or partition cells and an example is the set of at-zones of a grey-scale image I [SS95]. Representing each disjoint set requires a unique identifier which is often the set of coordinates or the index label of one of its elements. In the case of *linked lists* [Cor+09; Sed98; Sha98] this is the element at the head of the list, while in the case of *priority queues* [Cor+09; Tho07] this could be either the element at the head or the tail. Tarjan s [Tar75; Tar83] union-find algorithm uses a rooted-tree representation for disjoint sets. Each set is represented by a single tree with leaf nodes corresponding to the set elements. The identifier in this representation is the element defining the root of the tree, also referred to as the *root*, i.e. the one whose parent pointer points to itself.

Tarjan s method provides four basic operations for managing disjoint sets that are summarised in [WR00; MW02]. Revised for the needs of this work, the four methods, listed analytically in Alg. 1, are:

1. `MakeSet`$(\mathtt{p})$: Creates a new singleton set $\{\mathtt{p}\}$ from a non-labelled point p, i.e. a single-node tree;
2. `FindRoot`$(\mathtt{p})$: Returns the root of the tree containing p;
3. `Criterion`$(\mathtt{p},\mathtt{q},\alpha)$: Returns true if $\mathtt{d}(\mathtt{p},\mathtt{q}) \leq \alpha$;
4. `Union`$(\mathtt{p},\mathtt{q},\alpha)$: Merges the two trees containing p and q respectively, if the criterion returns true.

The dissimilarity measure is computed in a separate function that is custom to each application. In its simplest form, $\mathtt{d}(\mathtt{p},\mathtt{q})$ is the L_2 norm between p and q, i.e. $\mathtt{d}(\mathtt{p},\mathtt{q}) = ||\mathtt{Attr(p)} - \mathtt{Attr(q)}||$. This was used on grey-scale images by Wilkinson et al. and Meijster et al. in [WR00; MW02] respectively, for at zone labelling

and connected component attribute filtering. In both works `Attr(p)` was the image intensity at a point p, and α was implicitly assumed as 0.

In the α-tree algorithm discussed next, the disjoint sets that need to be labelled are α-CCs for which $\alpha \in [0,1,...,\text{NumLevels}-1]$. NumLevels is the maximum level of the input. For any α within this range the set of identifiers that accounts for the union-find forest of the image, is stored in an image-size-long array called `Parent[]`. For each non-root pixel p the entry `Parent[p]` is assigned a positive integer value which is the scan-line order offset of its parent with respect to the top-left corner of the image. If p is a root pixel, `Parent[p]` is set to -1 instead of pointing to itself. The negative sign indicates its root status while the number itself is utilised later as a node identifier.

4.2 The α-tree data structure

The α-tree data structure is a rooted, uni-directed tree, with its leaves corresponding to the reference connected components, i.e. the 0-CCs, of the input image, and the root to the tip of the α-hierarchy, i.e. the single connected component whose extent defines the image definition domain. Each tree node associates to a unique α-connected component of the hierarchy, and has a pointer to its parent. It is denoted as $\mathtt{N}_{\mathtt{k}(\alpha),\mathtt{i}}$, in which $k(\alpha)$ is the dissimilarity level mapped to a tree level, and i is the node index at $k(\alpha)$. The node parent associates with the first strict super set mapped into the hierarchy at level $k(\alpha')$ such that $k(\alpha') > k(\alpha)$. The node structure consists of four members - Fig. 4.1; the `offset`, the `parent`, the `new_offset` and the `attribute`. The `offset` is an integer variable storing the offset of the node s identifier pixel from the origin. The `parent` is an integer variable storing the parent node ID, and the `new_offset` is an integer that stores the new component identifier after segmentation or attribute filtering for the subsequent output restitution. The `attribute` variable is a void pointer to the `AuxiliaryData` structure. The latter is a data structure consisting of a set of generic parameters that are updated incrementally, i.e. for each pixel visited. This allows for a number of different attributes to be computed during the filtering stage from the very same set of auxiliary data.

The auxiliary data structure used along with the α-tree has been originally developed for the implementation of the Max-Tree algorithm presented in [MW02] and used in [WW01; OW07; WRW07; URW07]. It consists of a set of generic variables that are utilised in the computation of component attributes. The number of variables depends on the pool of attributes to be supported. Increasing it adds up to the memory requirement per node. The auxiliary data structure is manipulated by a set of data-handling functions. These are the `NewAuxData()` that pre-sets all members to an initial value, the `AddToAuxData()` that adds the contribution of a single pixel to each variable separately, the `MergeAuxData()` that given two pointers to the data structure, it computes the sum of the individual members and updates the first one, the `SetAuxData()` that sets each respective member to some given value, and the

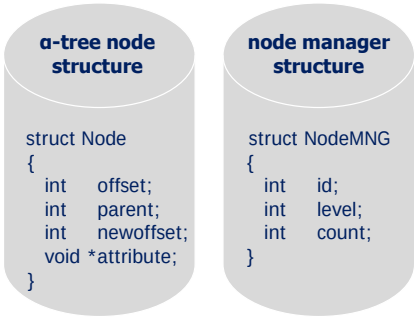


Fig. 4.1 The α-tree Node structure (left) and the NodeManager data structure (right).

`DeleteAuxData()` that frees the memory allocated for a given instance of the data structure.

The area attribute is computed trivially, and for non-compactness the method was described in [WW01]. The ω-range is computed from the *positive* and *negative range* members of the structure. These are initialised to the intensity of the connected component s identifier. For any other member pixel q of the α-CC under study, the positive range is updated with the maximum value between the existing one and `I[q]`, and the negative range with the minimum value between the existing one and `I[q]`. The component attribute is given by the absolute value of their difference. The variance and standard deviation attributes are computed in the same way as non-compactness only using intensities in place of the spatial coordinates.

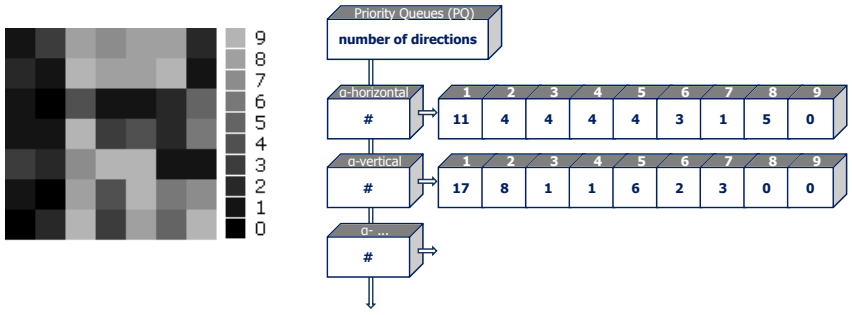
The α-tree construction consists of two phases; the initialisation and the tree population, presented next. The tree construction module makes use of a node managing system referred to as the *node manager*. The latter serves as a registry for the status of each propagating node at the population wavefront. A node can be `ACTIVE`, `STALLED` or `INACTIVE`. A node marked as `ACTIVE` registers itself on the α-tree structure and further propagates along with the population wavefront. A node marked as `STALLED` is already registered on the α-tree and propagates without leaving a trace on the tree until it meets its parent. When this occurs, it is marked as `INACTIVE` and stops propagating further.

4.3 The α-tree construction

The construction of the α-tree is achieved by the following two subsequent steps detailed in Secs. 4.3.1 and 4.3.2 respectively: (i) initialisation and labelling of the reference connected components (i.e., the 0-CCs); (ii) wavefront propagation from the reference connected components to the tip of the tree (i.e., once only the node representing the whole image domain is discovered).

4.3.1 *Initialisation and* 0-CC *Labelling*

The tree construction initiates by identifying and labelling all 0-CCs, which define the leaves of the tree. During initialisation, a pass through all image data is requested. For each pair of adjacent pixels their dissimilarity is computed and registered in a set of *dissimilarity histograms*. There exists one such histogram for each adjacency direction thus for the case of 4-connectivity two histograms are computed. For each dissimilarity histogram a priority queue is constructed. The set of priority queues is used for driving the tree population phase discussed in the next section. Note that since the pool of data is known a priory the priority queues are implemented in the form of hierarchical queues [Mey92; BM93; MW02], avoiding dynamic memory allocations. An example of two such hierarchical queues (HQueue) computed from Fig. 4.2a is shown in Fig. 4.2b.



(a) (b)

Fig. 4.2 (a) The test pattern of Fig. 3.1a in colour coding. (b) Two priority queues (HQueue) computed from (a) that register the number of adjacencies for each dissimilarity threshold α; one for the horizontal and the other for the vertical direction.

The initialisation function, listed in Alg. 2, is called with input parameters the α-tree `at`, the node manager `NodeMNG`, and a pointer to the image data. The `Parent`$[\#]$ array and the set of auxiliary-data-handling functions are members of the α-tree structure. For simplicity this analysis is limited to the 4-connectivity only. The function starts off with a forward scan through the image. Each pixel visited is set to define a reference component to which it also serves as a parent by calling `MakeSet`$(\mathtt{p})$. If a left neighbour `q` exists, a call to `Union`$(\mathtt{p},\mathtt{q},0)$ is issued. The dissimilarity measure between `p` and `q` is computed and if $\mathtt{d}(\mathtt{p},\mathtt{q}) = 0$, meaning that both points belong to the same reference component, a link is made by first finding the root r of `q` and setting `at.Parent`$[\mathtt{r}] = \mathtt{p}$. In this way, the last pixel processed is always set to be the root of the updated/new reference component. If $\mathtt{d}(\mathtt{p},\mathtt{q}) \neq 0$, the corresponding entry of the dissimilarity histogram is updated, i.e. `UpdateHistogram`$()$ in Alg. 2. The same routine is repeated in the presence of a neighbour above `p`. The statement $\mathtt{q} < \mathtt{p}$ means that the pixel `q` has been processed before `p`. The forward scan concludes after all pixels are visited. The `at.Parent`$[\#]$

Algorithm 2 Initialisation pseudo-code: union-find based 0-CC labelling and `HQueue` creation

```
Require: var at, NodeMNG, I // α-tree, node manager, input image.
 1: var p,q,dir; // pixel, neighbour, direction
 2: var NumActiveNodes = 0; // number of active nodes
 3: var node,size = SizeOf(Node); //α-tree node, size
 4: // Forward scan
 5: for p = 0 to ImageSize−1 step 1 do
 6:     MakeSet(p);
 7:     for all q = Get4neighbor(p) | q < p do
 8:         if Union(p,q,0)==false then
 9:             dir = GetDir(p,q);
10:             UpdateHistogram(d(p,q),dir);
11:             NumActiveNodes =UpdateNumNodes();
12:         end if
13:     end for
14: end for
15: for all dir do
16:     HQueueCreate(dir);
17: end for
18: at → Node = Alloc(2∗NumActiveNodes∗size);
19: NumActiveNodes = 0
20: // Backward scan - resolving pass
21: for p=ImageSize-1 to 0 step1 do
22:     if at → Parent[p]<0 then
23:         at → Parent[p] = -NumActiveNodes -1;
24:         SetNode(at → Node[NumActiveNodes]);
25:         UpdateNodeManager(NumActiveNodes);
26:         NumActiveNodes++;
27:     else
28:         at → Parent[p] = FindRoot(p);
29:         node = GetNodeFromParent[at → Parent[p]];
30:         AddToAuxData(node,p);
31:         for all q = Get4neighbor(p) | q > p do
32:             if Criterion(p,q,0)==false then
33:                 dir = GetDir(p,q);
34:                 HQueueAdd(dir,d(p,q),q);
35:             end if
36:         end for
37:     end if
38: end for
39: return NumActiveNodes;
```

array to this stage has registered a total of `NumActiveNodes` chained components that are subsets or full 0-CCs. This is an overestimate of the true number of disjoint sets. Assigning unique labels to the actual disjoint sets requires a backward scan, referred to as the "resolving pass" [MW02]. The initial estimate of `NumActiveNodes`, which is later corrected to the actual number of 0-CCs, is used for allocating system memory for the nodes of the leaf-level of the α-tree structure. Dealing with the worst case scenario in which each pixel differs from its left and top neighbour, it is required that each histogram has a total number of entries equal to half the total number of pixels. The tree size in this case equals to twice the number of 0-CCs. This is because, at each level $k(\alpha)$ of the hierarchy, there can be a maximal number of merges equal to half the number of nodes at the previous level $k(\alpha')$ for $\alpha' < \alpha$. The set of priority queues is created and initialised from the associated dissimilarity histograms by calling `HQueueCreate()`.

In the resolving pass through the image each pixel is visited in the reverse scanline order. If `p` is labelled as root it defines a tree-leaf node and updates the `NodeMNG`. Defining a node means associating a unique ID to an instance of the tree and setting its four members (Fig. 4.1-left) by calling `SetNode()`. This ID is given by the sum of the level offset, i.e. the total number of tree nodes prior to the given tree level $k(0)$ which is 0 in this case, and the number of active nodes registered so far at $k(0)$. The `offset` is set to `p`, the `parent` is initialised to the node ID, the `new_offset` is initialised to `offset`, and all members of the `AuxiliaryData` instance are set to their initial values. The entry `at.Parent[p]` is set to $-$`NumNodes` -1 such that each root is associated to a unique node ID. The `NodeMNG` data structure consists of three members - Fig. 4.1-right; the `id` that is initialised to `p`, the `level` set to the current α level, i.e. 0, and the `count` that is set to the current number of active nodes, i.e.

NumActiveNodes. Note that for $\alpha = 0$ all nodes are active nodes. If p is not a root pixel, its parent is retrieved by calling FindRoot(p). Pixels with non-root status update the auxiliary data of their root by calling the AddToAuxData() function. Additionally, for any p for which at.Parent[p] ≥ 0, if the dissimilarity measure computed between p and its right or bottom neighbour q, is greater than 0, then q is added to the corresponding HQueue entry by calling HQueueAdd(dir, d(p,q), q). At the end of this procedure, both queues are ooded and the leaf-level of the α-tree is completed.

4.3.2 Tree Population: The Wavefront Function

The tree population phase is realised through a wavefront propagation function that computes the union-find algorithm between α-CCs. It starts off from the first level of the α-hierarchy that is other than 0 and for which at least one node is present. It is an iterative top-down process that concludes after reaching the tip of the α-hierarchy. For each α value in the range $[1, 2, ..., \texttt{MaxNumLevels}]$ a call to CreateNewTreeLevel() is issued. The function receives at its input the α-tree at, the node manager NodeMNG, and the value of α to be processed. It returns the total number of active nodes for the given α. The pseudo-code is listed in Alg. 3.

The function initiates by accessing the set of vectors of the priority queue HQueue, using the value of α as argument. The members of each vector entry are accessed sequentially. For a member p its preceding neighbour q along the direction specified, is retrieved. The root elements of the components to which p and q belong to, are computed using FindRoot() and stored in par_p and par_q respectively. If the two differ, which means that the respective components have not been merged yet, the furthest one from the origin becomes the root of the new component. The α-CC containing it is addressed to as the *donor* component. The other, i.e. the component containing the pixel with the closest root to the origin, merges with the new component by redirecting its parent pointer to the donor parent.

If one or more new components are created the presence of a new tree level $k(\alpha)$ is confirmed. Each new component is marked as ACTIVE by adding a negative offset equal to the image size to at.Parent[new_root]. This is used for distinguishing ACTIVE from STALLED nodes, associated to the respective components. If no new tree level is confirmed the function returns the same number of active nodes for the given α as for $\alpha - 1$, i.e. NumActiveNodes is the same as NumPrevNodes. This indicates a redundant pyramid level [OS11]. The tree gets updated only when the two counts differ, and this happens after CreateNewTreeLevel() returns.

If a new tree level is confirmed, a resolving pass is required to assign unique labels to the new components and to set the corresponding nodes. The resolving pass launches a visit to all nodes registered at NodeMNG up until the previous tree level, from which the new nodes at $k(\alpha)$ will be constructed. The status of each node is determined from its associated component, i.e. by the respective signed value of the

Algorithm 3 Function: `CreateNewTreeLevel()` at threshold α

```
Require: at, NodeMNG, α, NumPrevNodes //α-tree, node manager
 1: var p,q,par_p,par_q; // pixel, neighbour, parents
 2: var level_offset,tmp; // level offset, temp. variable
 3: var c,l,n,dir; // counter, level, node counter, direction
 4: var IDX,TIDX; // node indices
 5: var NumActiveNodes,NumNewNodes = NumPrevNodes;
 6: // Forward pass through the CC edges at previous level
 7: for all dir do
 8:     while HQueueNotEmpty(dir,α) do
 9:         HQueueFirst(dir,α);
10:         q = Get4neighbor(p) | q < p ∧ dir==GetDir(p,q);
11:         par_p = FindRoot(p); par_q = FindRoot(q);
12:         if par_p != par_q then
13:             if par_p > par_q then
14:                 at→Parent[q]=par_p;
15:                 if at→Parent[par_p] > -ImageSize then
16:                     at→Parent[par_p] -= ImageSize;
17:                 end if
18:             else
19:                 at→Parent[p]=par_q;
20:                 if at→Parent[par_q] > -ImageSize then
21:                     at→Parent[par_q] -= ImageSize;
22:                 end if
23:             end if
24:             NumNewNodes- -;
25:         end if
26:     end while
27: end for
28: // Resolving pass through all nodes in node manager

29: if NumNewNodes < NumPrevNodes then
30:     NumActiveNodes = NumNodes = 0;
31:     level_offset = GetLevOffset(α);
32:     for var i = 0 to NumPrevNodes step 1 do
33:         if at→Parent[NodeMNG[i].id] < 0 then
34:             if at→Parent[NodeMNG[i].id] <-ImageSize then
35:                 // Active node: register it on the tree
36:                 IDX = level_offset + NumActiveNodes;
37:                 SetNode(at→Node[IDX]);
38:                 l=NodeMNG[i].level; c=NodeMNG[i].count;

39:                 TIDX = GetID(l,c);
40:                 at→Node[TIDX].parent = IDX;
41:                 SetAuxData(at,IDX,TIDX);
42:                 NodeMNG_RCFG(i,NumNodes,level);

43:                 NodeMNG[i].id = -NumActiveNodes-1;
44:                 NumActiveNodes++; NumNodes++;
45:             else
46:                 // Stalled node: reconfigure the node manager
47:                 NodeMNG_RCFG(i,NumNodes,level);

48:                 NumNodes++;
49:             end if
50:         else
51:             // Inactive node: finalise node
52:             tmp = FindRoot(NodeMNG[i].id);
53:             at→Parent[NodeMNG[i].id] = tmp;
54:             l=NodeMNG[i].level; c=NodeMNG[i].count;

55:             TIDX = GetID(l,c);
56:             tmp = at→Parent[NodeMNG[i].id];
57:             IDX = level_offset-1-at→Parent[tmp];
58:             at→Node[TIDX].parent = IDX;
59:             MergeAuxData(at,IDX,TIDX);
60:         end if
61:     end for
62: end if
63: return NumActiveNodes;
```

parent array. If greater or equal to 0, the node is `INACTIVE`. If less than `-ImageSize` the node is `ACTIVE`, and if between 0 and `-ImageSize`, it is `STALLED`.

A simple example, shown in Figs. 4.3 and 4.4, is used to demonstrate this process. The term $\mathtt{C}_{\alpha,\#}$ is used to indicate an indexed α-CC. It assumes that the process was concluded for α, and after the forward pass at α' with $\alpha < \alpha'$, the components $\mathtt{C}_{\alpha,3}$ and $\mathtt{C}_{\alpha,2}$ have been merged to $\mathtt{C}_{\alpha,1}$. This is indicated by the single plateau under them in Fig. 4.3. The component $\mathtt{C}_{\alpha,1}$ is the donor, i.e. its parent is the furthest away from the origin. The component $\mathtt{C}_{\alpha,0}$ was not modified thus associates to a `STALLED` node; it is a ghost replica propagating until it gets merged to some other component at a higher level.

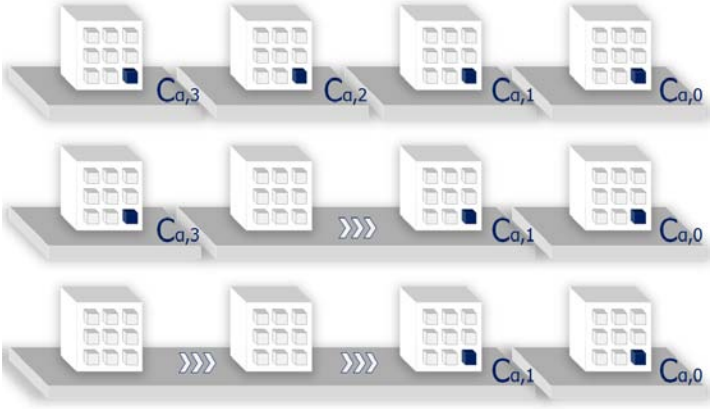


Fig. 4.3 An example of tree population: The forward pass is driven by components marked by pixels that are retrieved from the queue vectors at each given level.

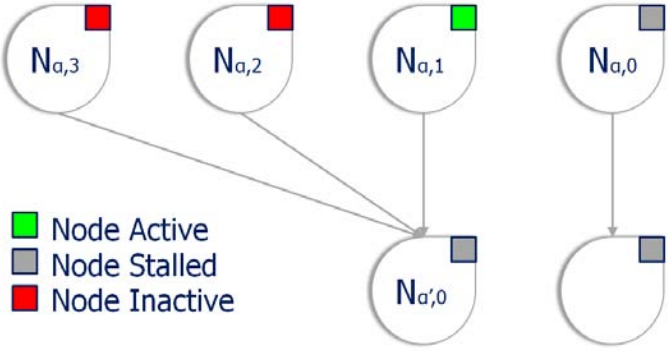


Fig. 4.4 An example of tree population: The forward pass visits the equivalent nodes and performs merging as requested.

The resolving pass starts off by reading the contents of the `NodeMNG` at level α. The identifiers of stored nodes are sorted in reverse scan-line order, thus reading the first entry retrieves the last node at tree level α. This is $\mathtt{N}_{\alpha,0}$. The process is shown in Fig. 4.4. Since $\mathtt{C}_{\alpha,0}$ was not modified during the forward pass, $\mathtt{N}_{\alpha,0}$ is marked as `STALLED` and only needs to update the respective entry of `NodeMNG` at α' in order to ensure its further propagation. This is done by calling the `NodeMNG_RCFG()` function in Alg. 3. The `NumNodes` counter is then updated. Note that the `NodeMNG` has initially as many entries as the number of 0-CCs, but reduces in size as α increases since the total number of nodes to be propagated reduces too. Each new level re-uses the already allocated memory with the `NodeMNG_RCFG()` overwriting on entries used for the previous level.

The next node from the right is $\mathtt{N}_{\alpha,1}$, which is marked as ACTIVE because $\mathtt{C}_{\alpha,1}$ was merged to other components and retained its original identifier. First, the node ID is set to the new tree level offset, i.e. the total number of nodes below the given value of α, plus the NumActiveNodes count until this instance. A new node $\mathtt{N}_{\alpha',0}$ is created and set by the SetNode() function in Alg. 3. The parent pointer of $\mathtt{N}_{\alpha,1}$, i.e. its direct descendant is set to the $\mathtt{N}_{\alpha',0}$ ID. The at.Parent[#] entry corresponding to the $\mathtt{N}_{\alpha',0}$ identifier is normalised by adding the $-$ImageSize offset to invalidate the ACTIVE status. A new status will be assigned at the next call to CreateNewTreeLevel(). Moreover, the NodeMNG reconfigures the members of its NumNewNodes entry according to the details of $\mathtt{N}_{\alpha',0}$. The NumNewNodes and NumActiveNodes counts are updated accordingly.

The remaining two nodes $\mathtt{N}_{\alpha,2}$ and $\mathtt{N}_{\alpha,3}$ are marked as INACTIVE. This is because the components $\mathtt{C}_{\alpha,2}$ and $\mathtt{C}_{\alpha,3}$ respectively, merged with $\mathtt{C}_{\alpha,1}$ and obtained a new identifier. The INACTIVE status means that they are excluded from propagating further using the NodeMNG. To finalise them the respective parent members need to be set. Calling the FindRoot() function returns the identifier of the component they have been merged to, i.e. $\mathtt{C}_{\alpha,1}$. This, in turn, is used to retrieve the node ID associated to this component. This is the ID of $\mathtt{N}_{\alpha',0}$ that has been already registered in the NodeMNG. The parent pointers of the two nodes are set to this ID and their respective auxiliary data are merged with those of $\mathtt{N}_{\alpha',0}$.

After conclusion of the resolving pass, CreateNewTreeLevel() returns the number of active nodes that are compared against the number of nodes of the previous tree level. If different, the new tree level is registered, the maximum number of nodes at this level is set to NumActiveNodes, and the total number of tree nodes before this level is updated for the next iteration of CreateNewTreeLevel(), when computing the new node IDs. The value of α is interpreted into a new tree level.

4.4 Tree Processing and Output Restitution

Operating directly on the nodes for computing image filters or the segmentation transform, requires two passes through the tree structure; a forward pass for invalidating nodes according to the Max-Rule discussed in Sec. 3.4 and a second, inverse pass for preparing the output.

The forward pass initiates at α=1 since the 0-CCs are not subject to any criterion, and terminates at the value defining the upper bound of the hierarchy. The nodes present at each plane of the hierarchy are accessed sequentially and each one separately is subjected to the attribute criterion, complemented (logical **OR**) by a further check on its status ag. If either the node s attribute is greater than the given threshold or the Status ag is invalidated, the status ag of both the node under study and its parent are invalidated. This ensures that once a node is invalidated, this decision propagates all the way to the end of the root-path.

The inverse pass through the tree structure initiates from the root and terminates after processing the leaves. For each node accessed, the status ag of its parent

is inspected. If valid, i.e. the parent satisfies the criterion, the `Offset` member of the node under study is set to that of its parent s. This way, as the partition becomes finer, i.e. we move down the α range, there is only one identifier that is propagated to all the ascendants. This is called the identifier of the *donor* node.

The output-image restitution requires a single pass through a replica of the 0-CC labelled image. If `Parent`[`p`] < 0, i.e. `p` is an identifier, the ID of the donor node is computed as -`Parent`[`p`] - 1. This yields the node offset at the 0-CC level. Alternatively, if `Parent`[`p`] > 0 the ID is computed as -`Parent`[`Parent`[`p`]] - 1. The output value at `p` can be set to the label/ID of the node it belongs to, or to some other grey-scale or colour based representation.

Chapter 5
Computational Performance

The design of the α-tree algorithm was motivated by three main directives; the ability to compute the entire unconstrained hierarchy of partitions from a given image based on a pre-selected dissimilarity measure, the ability to set dynamically the attribute based on which the respective constraints are enforced, and the ability to produce a segmentation of the represented image interactively and in near real-time.

All three objectives were tested on a series of experiments using a massive, very high resolution (VHR) satellite image tile of Beirut, Lebanon. The original image is a 256 megapixel WorldView-2 RGB acquisition whose luminance channel is shown in Fig. 5.1. It is separated in three regions of decreasing scene complexity, starting from the right. Each region is colour-shaded for display purposes only. The first region corresponds to urban and partially vegetated regions, the second to partially urban regions adjacent to the sea front, and the third to regions into the sea. Experiments carried out with respect to variable image size, used a series of sub-tiles of the full image in Fig. 5.1, ordered with respect to increasing size and decreasing scene complexity. The dissimilarity measure for all experiments was the L_2 norm. All experiments were carried out on a dual quad-core Intel Xeon CPU (E5504)@2GHz with 16 GB RAM.

The correctness of the segmentation procedure was tested by comparing the results of the respective α-tree module against the (α,ω)-segmentation algorithm [Soi08]. The experiment was launched on the full image shown in Fig. 5.1 for the three attributes supported in common by both implementations; the area, ω-range and variance. The comparison was realised by computing the goodness-of-fit between pairs of corresponding partition cells, obtained from the two algorithms. The comparison converged to a complete match for all tested cells.

The first experiment tested the suitability of both methods for computing an unconstrained hierarchy. Timings for each hierarchy level are given in Fig. 5.2a. The image used for this test was a $4,096^2$-element (i.e., 16 megapixels) sub-tile with its top right corner coinciding with the point (0,0) in Fig. 5.1. In this task, the (α,ω)-segmentation algorithm needs to be iterated for each value of α and to have ω set to the maximum intensity range. The corresponding curve shows an almost increasing time requirement that is due to the algorithm s dependency on previously computed

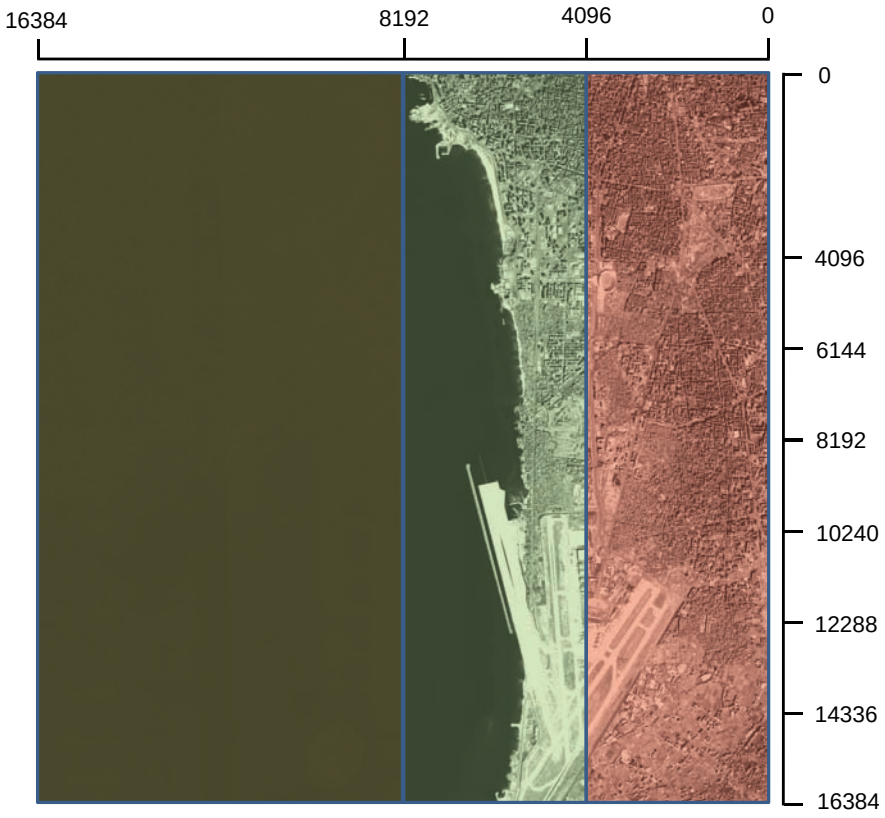
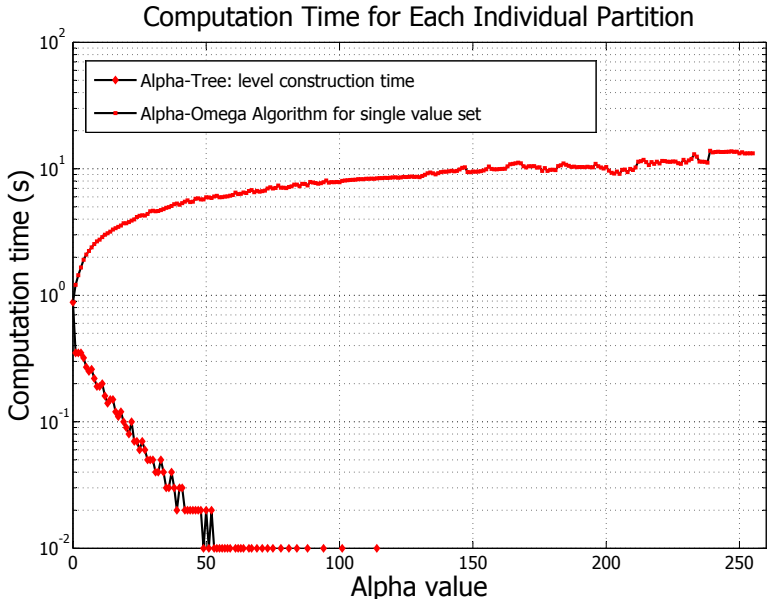


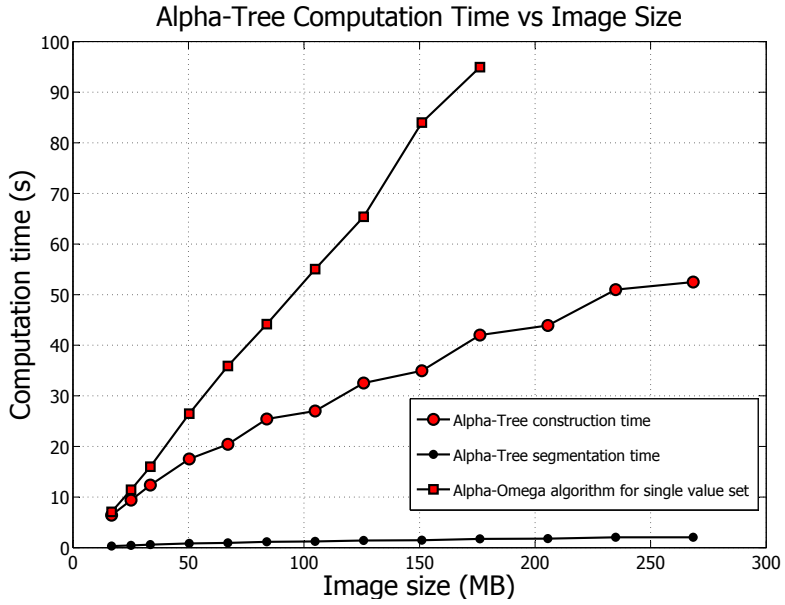
Fig. 5.1 The 256M test image with three regions of decreasing scene complexity marked with colour shades.

partitions, i.e. for each hierarchy level α, all other partitions for $\alpha' < \alpha$ need to be computed first. This shows a sharp increase for small values of α due to the big number of merges, and smoothes out as α increases further. By contrast the α-tree algorithm has a progressive decline in the time requirement because each hierarchy level is computed directly from the previous hierarchy level. Consequently, the number of merges reduces sharply. Moreover, the α-tree algorithm considers only partitions cells of the input image that differ from their predecessors. In this example the α-tree structure consisted of **108** hierarchy levels, with a total of **19,131,284** nodes and was computed in **6.52s** The (α, ω)-segmentation algorithm computed for **256** values of α was concluded after **2138.42s**.

The second experiment tested the computation time of both algorithms as a function of image size. The experiment was computed for an (α, ω) set of values such that α=ω=80. The results are plotted in Fig. 5.2b. Both algorithms have a rather smooth and almost linear response with respect to image size, for the first 10 subtiles. The (α, ω)-segmentation algorithm however, shows a dependence on large

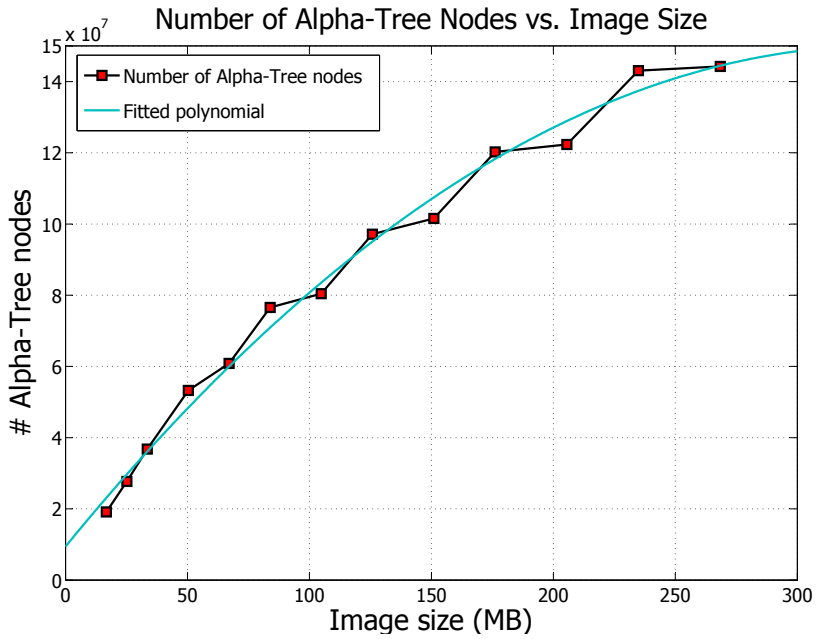


(a)

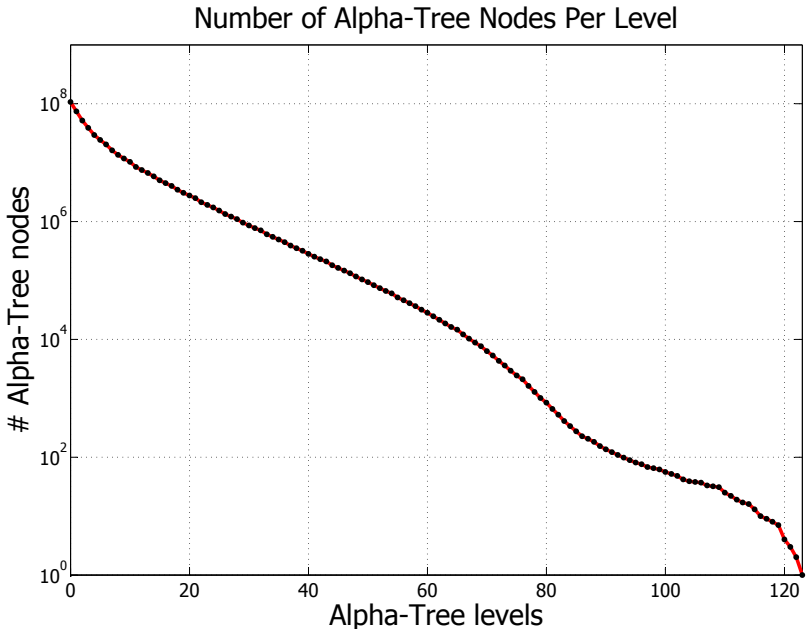


(b)

Fig. 5.2 α-tree performance graphs: (a) Timings for the constructing an (α,ω)-hierarchy based on the α-tree and the regular (α,ω)-segmentation algorithm for the $4,096^2$ sub-tile of the image shown in Fig. 5.1. (b) The α-tree construction and segmentation time against the (α,ω)-segmentation algorithm, as a function of increasing image sub-tile size.



(a)



(b)

Fig. 5.3 α-tree performance graphs: The number of tree nodes as a function of increasing image sub-tile size, and of tree levels (logarithmic scale), in (a) and (b) respectively. In (b), the whole 256 megapixel image shown in Fig. 5.1 was used.

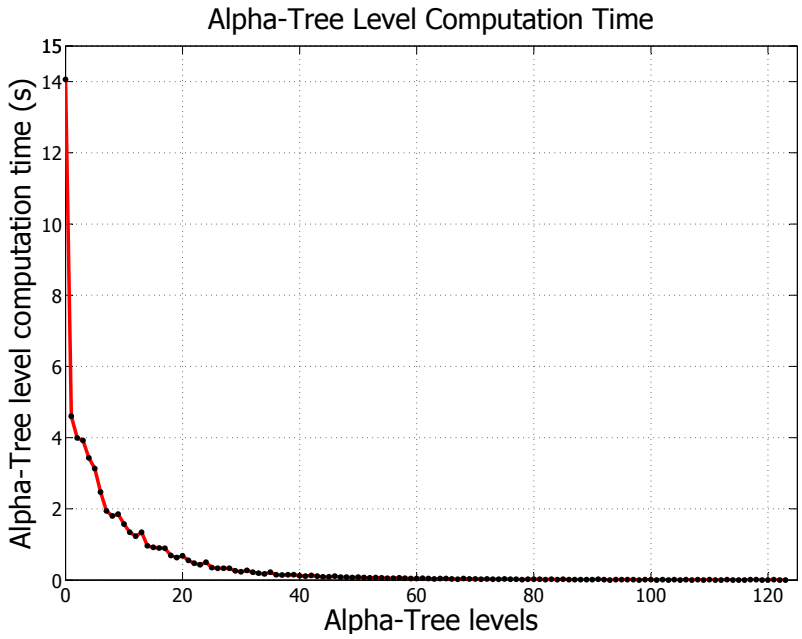


Fig. 5.4 α-tree performance graph: The computation time of each α-tree level for the whole 256 megapixel test image of Fig. 5.1.

image regions of rather smooth intensity variation, and for the last three image sub-tiles including most of the sea component, it concludes after **2,188.39s**, **2,936.93s**, and **9,959.05s** respectively. These timings are not shown in Fig. 5.2b. A linear fit on the (α,ω)-segmentation is modelled by $y = 0.58x - 3.9$, and is computed from the first 10 values, since the last three lead to inconclusive results. A linear fit on the α-tree algorithm is modelled by $y = 0.18x + 7$. Extrapolating the computation time for image size equal to 1GB, this gives **576s** and **187s** respectively. Studying the computation time of α-tree-algorithm, a progressive reduction in the rate of time increase is observed for higher image size. This is due to the reduced scene complexity, since larger regions are described by disproportionately lesser number of nodes. This is more evident at the last three image sub-tiles where large segments of the sea component are included. The segmentation time for the largest size value is **2.05s** suggesting that the near real-time interaction with the tree structure is possible.

The third experiment reported in Fig. 5.3a confirms the α-tree s dependency on the image content by measuring the number of tree nodes as a function of image size. The fitted polynomial function has sharper rate of tree node reduction for big images containing larger sea components.

The last two experiments are computed on the largest image tile of Fig. 5.1 and measure the number of α-tree nodes and α-tree level computation time as a function of the tree level. The results are shown in images Figs. 5.3b and Fig. 5.4 respectively.

The number of tree nodes, plotted in logarithmic scale shows a rapid reduction as the value of tree level increases. This is expected since as we move towards the root of the tree, larger areas are grouped into bigger connected components and the number of merges drops accordingly. This has a similar effect on the computation time of each level (Fig. 5.4).

Chapter 6
Application Examples

The α-tree algorithm finds usage in image segmentation and simplification as well as object detection applications that require the exploitation of the full hierarchy of α-connected components. Once the tree is constructed, the speed of the tree processing and image restitution steps allows for interactive fine tuning if necessary.

This chapter presents a series of applications heavily relying on the α-tree algorithm. They all deal with the processing of VHR aerial or satellite images. Section 6.1 demonstrates the use of α-tree for the automatic extraction of connected components satisfying a series of constraints applied to shape, size, and spectral attributes. The use of interactive image information mining based on a switchboard platform based on the pattern spectrum of the connected components mapped by the α-tree is illustrated in Sec. 6.2.

6.1 Automatic Building Footprint Detection

Human settlements and in particular urban footprints provide a valuable input in all aspects of crisis management, ranging for disaster risk reduction and preparedness to post-disaster needs assessments. This information is required at a global scale and until recently it was only available at low to medium spatial resolution. Examples are the night-lights series [Elv+01] and the MODIS "urban layer" [SFP10]. The JRC has demonstrated through the release of the first prototype of the high resolution Global Human Settlement Layer (GHSL), the capacity to extract built-up information fully automatically and in a globally consistent manner from heterogeneous optical data sources. The GHSL is publicly available at `http://ghslsys.jrc.ec.europa.eu` and it is displayed at the OSGEO TMS scale 11. This corresponds to approximately 75m of spatial resolution at the equator. The internal GHSL release offers the global mosaic at TMS scale 14 corresponding to approximately 9.5m at the equator.

The GHSL is populated from a sophisticated engine based on evolutionary systems and advanced morphological image processing modules. The latter rely on

panchromatic or single-band inputs that offer a robust discrimination of built-up at the targeted spatial resolution (2.5m in the current version). Increasing the spatial resolution further and reaching the sub-meter domain requires enhanced approaches that make the best use of multi-spectral information while providing information on the building footprints and/or their substructures. The α-tree representation and the subsequent attribute constrained filtering and image restitution fulfil these requirements. Indeed, hierarchical segmentation based on constrained connectivity yields clusters of connected components representing candidate building footprints as already suggested in [Soi10; PS12]. In these publications, fixed total range attribute (ω) values were used. This approach is equivalent to performing non-horizontal cuts in the α-tree where the largest connected components satisfying the total range attribute constraint are extracted. The resulting connected components produce a partition of the image domain. Variable thresholds adapting to the local image structures can be obtained by analysing the persistence (lifetime) of the nodes of the dendrogram representing a hierarchy of constrained connected components as proposed in [Soi08, Sec. 4.2]. The α-tree algorithm presented in this report enables the efficient selection of α-tree nodes satisfying any given series of constraints and therefore exploits fully the richness of the underlying image partition hierarchy. For example, in the case of buildings as observed in VHR images, appropriate attributes with their associated constraints on the attribute values can be used to extract a mask of candidate building footprints. This approach is illustrated in Fig. 6.1.

The input image shown in Fig. 6.1a is a subset of a pan-sharpened true colour QuickBird image of the city of Sana a, Yemen with a spatial resolution of 60cm. A close-up view is shown in Fig. 6.1b. A section of the α-tree of Fig. 6.1b is shown in image Fig. 6.1c, in which the centre points of the big concentric circles of the spatially organised nodes, correspond to α-CCs registering a massive component merging. The image in Fig. 6.1d shows the partition of Fig. 6.1b into the reference 0-CCs using a dissimilarity defined as the maximum of the dissimilarities calculated for each colour channel separately. For each channel, the dissimilarity taking into account the neighbourhood of each pair of adjacent pixels was considered, see [Soi11] for details. The connected components matching the highest tree nodes satisfying a series of constraints related to the size, shape, and spectral properties of potential building footprints are shown in Fig. 6.1e for the close-up view and in Fig. 6.1g for the input image. Shape constraints are based on criteria such as elongation, rectilinearity [ZR03], and non-compactness [URW07].

Once a mask of the building footprints is calculated, higher level semantics such as building heights to achieve 3D urban scene reconstruction can be derived. Height information is computed from object shadows as a function of the sun elevation and azimuth angles. Figure 6.2a is a subset of a RGB VHR satellite image with a spatial resolution of 50cm. The partition shown in Fig. 6.2b corresponds to a non-horizontal cut of the α-tree for fixed parameters (α,ω)=(400,400). The mask of potential building footprints shown in Fig. 6.2c is obtained with a procedure similar to that described in the Sana a case study described previously (i.e., selection of nodes of the α-tree satisfying a series of shape, size, and spectral constraints). Preserved connected components have height information estimated from the length of

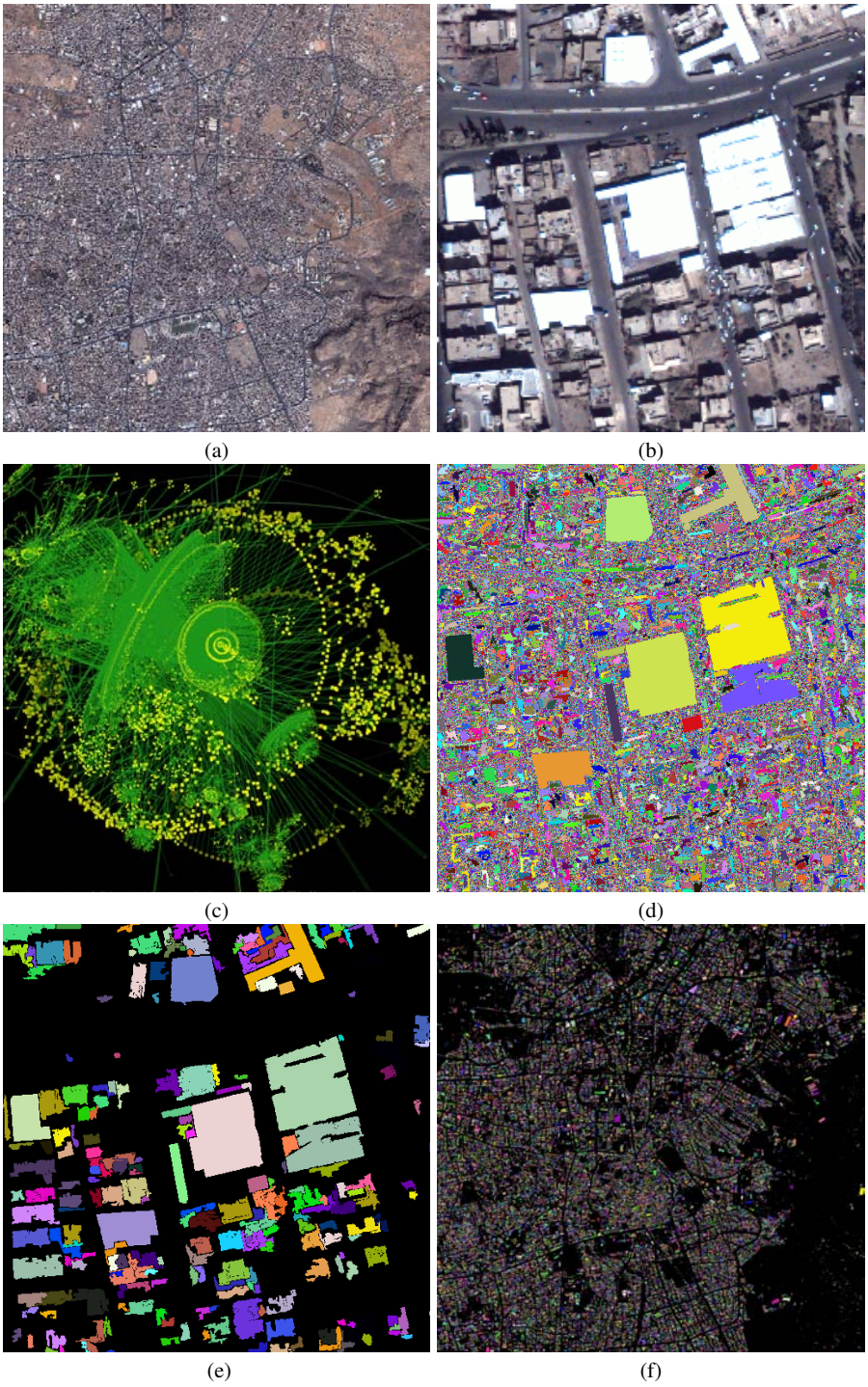


Fig. 6.1 (a–b) The input VHR satellite image and a zoom-in section. (c) A section of the α-tree in fish-eye view. (d) The partition into reference connected components corresponding to the 0-CCs. (e–f) The largest connected components of the tree satisfying a series of constraints (result of tree processing and image restitution) on the zoom-in section and the input VHR image.

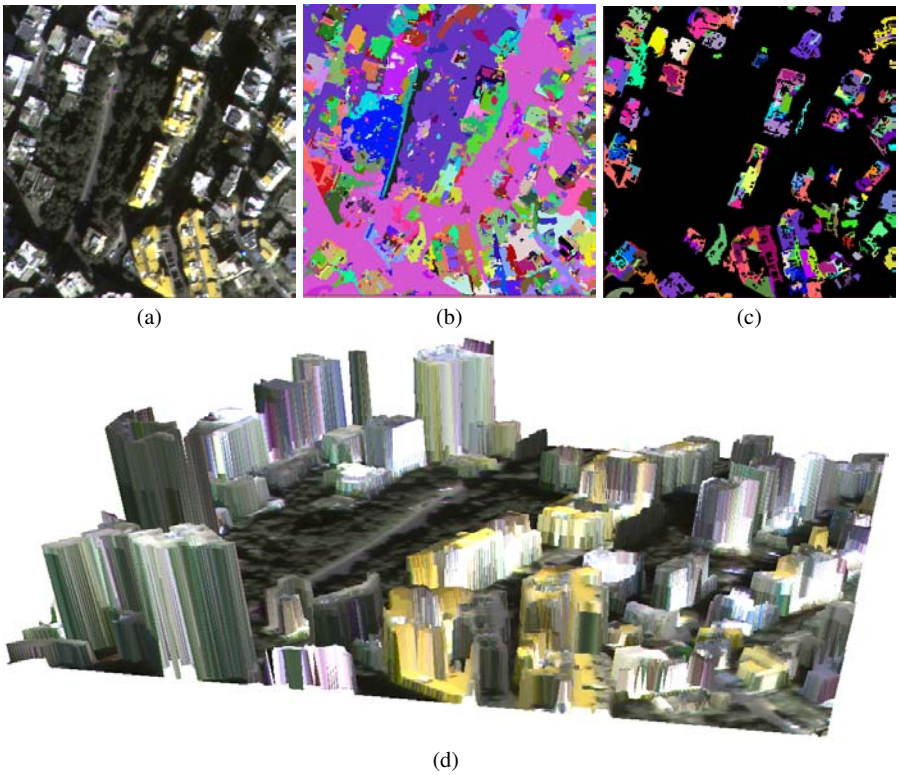


Fig. 6.2 3D urban scene reconstruction: (a) original image; (b) restitution of the largest connected components of the α-tree based on α,ω predicates; (c) connected components of the α-tree satisfying a series of size, shape, and spectral constraints; (d) the output rendered scene with building heights estimated from shadow lengths.

the shadows their corresponding building are casting and given the a priori known sun elevation and azimuth angles. This height information is imprinted as shades of the grey-scale from which the 3D rendered scene, shown in Fig. 6.2d, is computed.

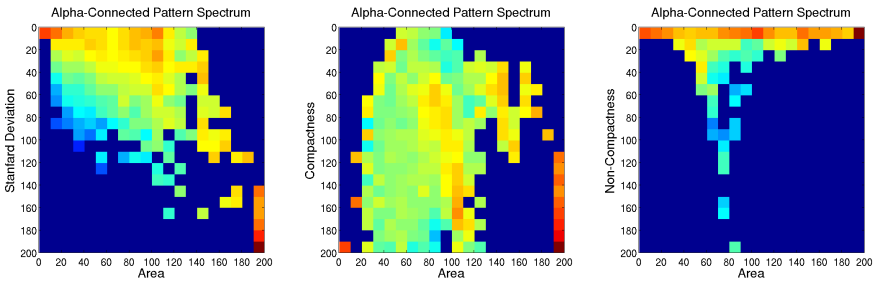
6.2 Interactive Image Information Mining

Interactive image information mining based on the α-tree representation can be implemented through machine learning approaches where the selection criteria for producing the desired connected components are automatically learnt from positive and negative examples selected by the user. This approach is detailed in [Gue+12].

Alternatively, rather than learning automatically the discriminant attributes and their associated constraints, user interaction with the image and a series of 2-dimensional scatterograms can be used to either display all those connected com-

ponents of the tree that are contributing to user selected points of the scatterogram or, conversely, highlight on the scatterogram the points matching α-connected components falling within a user-selected region of the image domain. This approach considering the scatterogram as a *switchboard* was introduced in [OS11] and developed in [Ouz+12]. The proposed 2-dimensional scatterograms are called 2-dimensional α-connected component pattern spectra. The value at a given position of a 2-dimensional pattern spectrum corresponding to a given pair of attributes indicates the energy contribution of the α-connected components of the α-tree whose attribute values fall within the ranges of values defined by this position. These ranges are set by a predefined binning function. The energy of a connected-component is defined as its area multiplied by its persistence, i.e., the interval of α-values for which it exists. For example, let us consider the area and standard deviation attributes and the binning functions that proceeds by multiples of 10 pixels and 20 intensity levels respectively. With this setting, the value at the coordinates $(7,2)$ represents the sum of the product of the area and persistence of the connected components whose area falls in the interval $[60,70[$ *and* whose standard deviation value falls in the interval $[20,40[$.

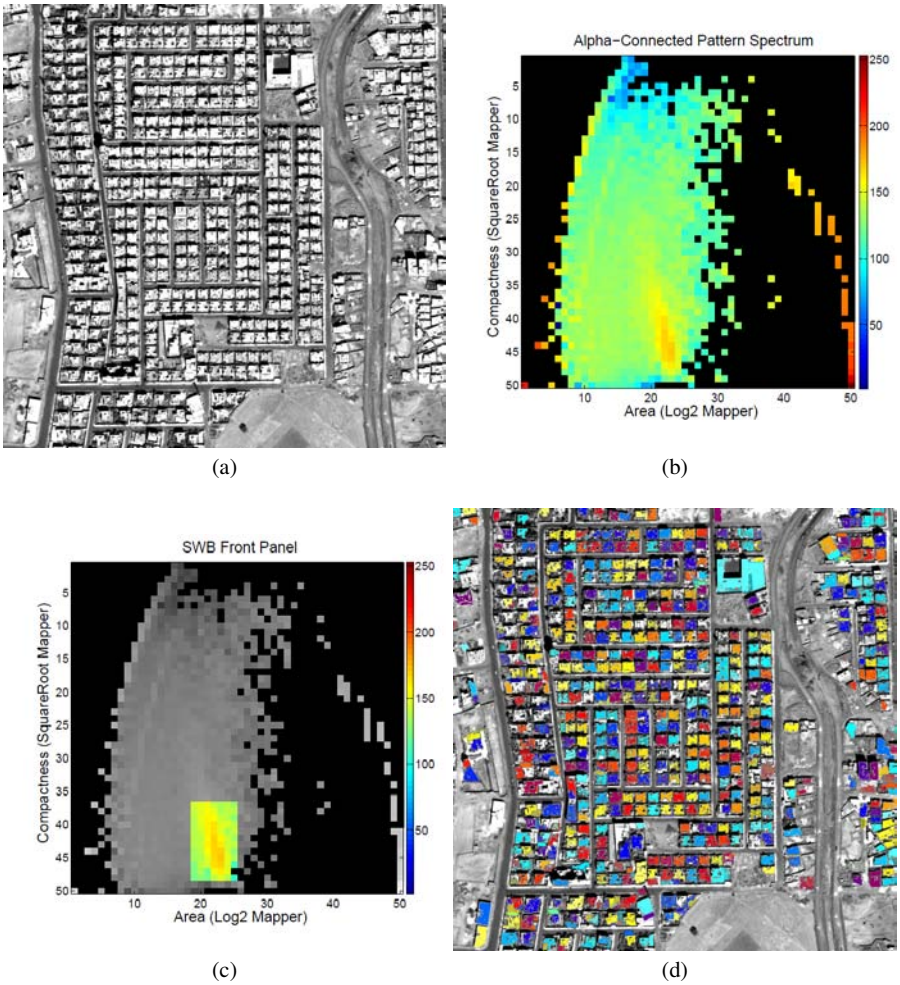
Figure 6.3 shows three different 2D α-connected pattern spectra using the area of α-CCs as a reference attribute and against standard deviation, compactness, and non-compactness attributes in (a), (b), and (c) respectively. The distribution of the image content with respect to each individual attribute differs considerably from the rest, suggesting that there is no single optimal metric but instead it is application dependent.



(a) (b) (c)

Fig. 6.3 Examples of α-connected pattern spectra for component area against standard deviation, compactness, and non-compactness in (a), (b), and (c) respectively.

Control mechanisms interfacing the α-connected pattern spectrum like the switchboard in [Ouz+12], offer a set of rich functionalities for interactive and rapid image information mining. The example of Fig. 6.4 shows a view of a residential area in the city of Sana a, Yemen. The test image is a subset of a QuickBird panchromatic acquisition. The original image has a spatial resolution of 60cm and is quantised to 8 bits/pixel. Aiming at building extraction for producing footprint maps, the system presented in [Ouz+12] learns the best describing target attributes and threshold



(a) (b)

(c) (d)

Fig. 6.4 Example of built-up extraction using α-connected pattern spectra as switchboard: (a) the test image; (b) the α-connected 2D pattern spectrum with area mapped on the horizontal axis and compactness on the vertical axis; (c) the switchboard (SWB) front panel with selected pattern-spectrum bins that produce the result in (d).

ranges from a set of positive examples identified manually. These are back-projected to the user on top of the pattern-spectrum. The latter is shown in image (b). The binning functions used in this example are the logarithm in base 2 and the square root of the attribute values for the area and compactness attributes respectively. In each case the 50 bins are linearly distributed on the range of the mapped values. The user, guided by this visual analytics platform can then set groups of, or individual switches that instruct the system on the specification of objects to be retrieved. A cluster of switches set for the particular example is shown in image (c). The system

scans through the α-tree structure and returns all the α-CCs satisfying this specification. The resulting components are associated with a random colour and are overlaid on the original image in image (d).

Chapter 7
Conclusion

A new method for hierarchical segmentation based on attribute constrained connectivity relations was presented. The framework introduced in this work has several desirable properties such as a strict nesting order of consecutive partitions subject to a dissimilarity metric, linkage order independence, and prevention of leakage through transitions. It does not require a model approach and delivers an unconstrained hierarchy of partition cells that spans over the entire range of values of a given dissimilarity metric. The full extent of this hierarchy is mapped on to the α-tree structure which is essentially a compact representation of the stack of α-connected partitions. By contrast to the Min-Tree of the edge graph that does not encode the reference components leading in certain cases to a notion of α-hypoconnection, the α-tree is a complete representation accounting for all values of α, with nodes strictly associated to unique α-connected components.

An efficient algorithm for computing the α-tree structure and constraining operators was presented. The α-tree is computed directly from the raw image data through an advanced connected component labelling scheme utilising Tarjan s union-find method. The algorithm accesses pixels in a scan line order and can handle multiple connected components simultaneously. During this process, α-CCs are created and merged as needed. It has three distinctive features; it computes the partition hierarchy efficiently, it allows for dynamic re-adjustment of the attribute used for enforcing constraints and it allows for interactive, near real-time segmentation of the input image. The performance of the α-tree algorithm was tested with respect to a number of different parameters and was compared against the original (α, ω) segmentation algorithm of [Soi08]. The results indicate a considerable improvement in performance, with the α-tree maintaining a lead in terms of exibility and computation time. Application examples were given related to segmentation of remote sensing imagery.

The α-tree algorithm further to image segmentation supports a wider family of image operators on graphs. Examples are attribute filters for image simplification and the computation of pattern spectra from partition-hierarchies. The practical usage of such tools was demonstrated in Chap. 6.

In future work we aim at introducing dissimilarity measures computed from hyper-spectral imagery and delivering a parallel implementation of the existing algorithm benefiting from related computational advances [AAY10]. The use of the α-tree structure for modelling partition hierarchies of non image-related data sources is currently being investigated.

References

[AAY10] P. Agarwal, L. Arge, and K. Yi. "I/O-efficient batched union-find and its applications to terrain analysis". *ACM Transactions on Algorithms* 7.1 (Dec. 2010), 11:1–11:21. DOI: 10.1145/1868237.1868249 (cit. on p. 44).

[And02] P. Andrews. *An Introduction to Mathematical Logic and Type Theory: To Truth Through Proof*. 2nd ed. Kluwer Academic Publishers, 2002 (cit. on p. 14).

[Bar77] J. Barwise. "An introduction to first-order logic". In: *Handbook of Mathematical Logic*. Ed. by J. Barwise. Vol. 90. Studies in Logic and the Foundations of Mathematics. Amsterdam: North-Holland Publications, 1977, pp. 5–46. DOI: 10.1016/S0049-237X(08)71097-8 (cit. on p. 14).

[BG89] J.-M. Beaulieu and M. Goldberg. "Hierarchy in picture segmentation: a stepwise optimization approach". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 11.2 (June 1989), pp. 150–163. DOI: 10.1109/34.16711 (cit. on p. 2).

[Ber+07] Ch. Berger, T. Géraud, R. Levillain, N. Widynski, A. Baillard, and E. Bertin. "Effective component tree computation with application to pattern recognition in astronomical imaging". In: *IEEE International Conference on Image Processing (ICIP)*. Vol. 4. Sept. 2007, pp. 41–44. DOI: 10.1109/ICIP.2007.4379949 (cit. on p. 17).

[BM93] S. Beucher and F. Meyer. "The morphological approach to segmentation: the watershed transformation". In: *Mathematical Morphology in Image Processing*. Ed. by E. Dougherty. Vol. 34. Optical Engineering. New York: Marcel Dekker, 1993. Chap. 12, pp. 433–481 (cit. on p. 21).

[BG02] U. Braga-Neto and J. Goutsias. "Connectivity on complete lattices: new results". *Computer Vision and Image Understanding* 85.1 (2002), pp. 22–53. DOI: 10.1006/cviu.2002.0961 (cit. on p. 6).

[BJ96] E. Breen and R. Jones. "Attribute openings, thinnings and granulometries". *Computer Vision and Image Understanding* 64.3 (1996), pp. 377–389. DOI: 10.1006/cviu.1996.0066 (cit. on pp. 1, 6).

[Cor+09] T. Cormen, Ch. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. 3rd ed. MIT Press, 2009 (cit. on p. 18).

[DST92] M. Dillencourt, H. Samet, and M. Tamminen. "A general approach to connected-component labeling for arbitrary image representations". *Journal of the ACM* 39.2 (Apr. 1992), pp. 253–280. DOI: 10.1145/128749.128750 (cit. on p. 17).

[Elv+01] C. Elvidge, M. Imhoff, K. Baugh, V. Hobson, I. Nelson, and J. Safran. "Nighttime lights of the world: 1994–1995". *ISPRS Journal of Photogrammetry and Remote Sensing* 56.2 (2001), pp. 81–99. DOI: 10.1016/S0924-2716(01)00040-5 (cit. on p. 35).

[FG96] C. Fiorio and J. Gustedt. "Two linear time union-find strategies for image processing". *Theoretical Computer Science* 154 (1996), pp. 165–181. DOI: doi:10.1016/0304-3975(94)00262-2 (cit. on p. 17).

[Gér05] T. Géraud. "Ruminations on Tarjan s union-find algorithm and connected operators". In: *Proc. ISMM'05*. Ed. by C. Ronse, L. Najman, and E. Decenciere. Computational Imaging and Vision. Springer, 2005, pp. 105–116. DOI: 10.1007/1-4020-3443-1_11 (cit. on p. 17).

[GR69] J. Gower and G. Ross. "Minimum spanning trees and single linkage cluster analysis". *Applied Statistics* 18.1 (1969), pp. 54–64. URL: http://www.jstor.org/stable/2346439 (cit. on p. 2).

[Gue+12] L. Gueguen, G.K. Ouzounis, M. Pesaresi, and P. Soille. "Tree based representations for fast information mining from VHR images". In: *Proc. of 8th Conference on Image Information Mining*. European Commission, Joint Research Centre, Oct. 2012, pp. 15–20. DOI: 10.2788/49465 (cit. on p. 38).

[GS11] L. Gueguen and P. Soille. "Frequent and dependent connectivities". In: *Proc. 10th Int. Symp. Math. Morphology (ISMM) 2011*. Vol. 6671. Lecture Notes in Computer Science. 2011, pp. 120–131. DOI: 10.1007/978-3-642-21569-8_11 (cit. on pp. 7, 14).

[HK03] Y. Haxhimusa and W. Kropatsch. "Hierarchy of partitions with dual graph contractions". *Lecture Notes in Computer Science* 2781 (2003), pp. 338–345. DOI: 10.1007/978-3-540-45243-0_44 (cit. on p. 2).

[Hes03] W. Hesselink. "Salembier s Min-tree algorithm turned into breadth first search". *Information Processing Letters* 88 (2003), pp. 225–229. DOI: 10.1016/j.ipl.2003.08.003 (cit. on p. 17).

[HP76] S. Horowitz and T. Pavlidis. "Picture segmentation by a tree traversal algorithm". *Journal of the ACM* 23.2 (Apr. 1976), pp. 368–388. DOI: 10.1145/321941.321956 (cit. on p. 1).

[Jon97] R. Jones. "Component trees for image filtering and segmentation". In: *Proc. of IEEE Workshop on Nonlinear Signal and Image Processing (NISP)*. Ed. by E. Coyle. Mackinac Island, Sept. 1997. URL: http://www.iwaenc.org/proceedings/1997/nsip97/pdf/scan/ns970311.pdf (cit. on pp. 1, 17).

[Jon99] R. Jones. “Connected filtering and segmentation using component trees”. *Computer Vision and Image Understanding* 75.3 (1999), pp. 215–228. DOI: 10.1006/cviu.1999.0777 (cit. on pp. 1, 17).

[Kru+93] F. Kruse, A. Lefkoff, J. Boardman, K. Heidebrecht, A. Shapiro, P. Barloon, and A. Goetz. “The spectral image processing system (SIPS)—Interactive visualization and analysis of imaging spectrometer data”. *Remote Sensing of Environment* 44.2–3 (May 1993), pp. 145–163. DOI: 10.1016/0034-4257(93)90013-N (cit. on p. 7).

[MW02] A. Meijster and M.H.F. Wilkinson. “A comparison of algorithms for connected set openings and closings”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 24.4 (2002), pp. 484–494. DOI: 10.1109/34.993556 (cit. on pp. 3, 17–19, 21, 22).

[Mey92] F. Meyer. “Integrals and gradients of images”. In: *Image Algebra and Morphological Image Processing III*. Ed. by P. Gader, E. Dougherty, and J. Serra. Vol. SPIE-1769. 1992, pp. 200–211. DOI: 10.1117/12.60643 (cit. on p. 21).

[Mey94] F. Meyer. “Minimum spanning forests for morphological segmentation”. In: *Mathematical Morphology and its Applications to Image Processing*. Ed. by J. Serra and P. Soille. Kluwer Academic Publishers, 1994, pp. 77–84 (cit. on p. 2).

[MM00] F. Meyer and P. Maragos. “Nonlinear scale-space representation with morphological levelings”. *Journal of Visual Communication and Image Representation* 11.3 (2000), pp. 245–265. DOI: 10.1006/jvci.1999.0447 (cit. on pp. 2, 8).

[MLC86] O. Morris, M. Lee, and A. Constantinides. “Graph theory for image analysis: an approach based on the shortest spanning tree”. *IEE proceedings* 133.2 (1986), pp. 146–152. DOI: 10.1049/ip-f-1:19860025 (cit. on p. 2).

[Nac95] P. Nacken. “Image segmentation by connectivity preserving relinking in hierarchical graph structures”. *Pattern Recognition* 28.6 (1995), pp. 907–920. DOI: 10.1016/0031-3203(94)00172-I (cit. on p. 2).

[NMI79] M. Nagao, T. Matsuyama, and Y. Ikeda. “Region extraction and shape analysis in aerial photographs”. *Computer Graphics and Image Processing* 10.3 (1979), pp. 195–203. DOI: 10.1016/0146-664X(79)90001-7 (cit. on pp. 2, 8).

[Naj09] L. Najman. “Ultrametric watersheds”. In: *Proc. Int. Symp. Math. Morphology (ISMM) 2009*. Vol. 5720. Lecture Notes in Computer Science. 2009, pp. 181–192. DOI: 10.1007/978-3-642-03613-2_17 (cit. on pp. 2, 12).

[Naj11] L. Najman. “On the equivalence between hierarchical segmentations and ultrametric watersheds”. *Journal of Mathematical Imaging and Vision* 40.3 (2011), pp. 231–247. DOI: 10.1007/s10851-011-0259-1 (cit. on p. 12).

[NC06] L. Najman and M. Couprie. “Building the component tree in quasi-linear time”. *IEEE Transactions on Image Processing* 15.11 (Nov. 2006), pp. 3531–3539. DOI: 10.1109/TIP.2006.877518 (cit. on p. 17).

[NS10] L. Najman and P. Soille. “On morphological hierarchical representations for image processing and spatial data”. In: *Proc. of ICPR Workshop on Applications of Discrete Geometry and Mathematical Morphology*. Ed. by U. Koethe, A. Montanvert, and P. Soille. Istanbul: IAPR, Aug. 2010, pp. 52–61 (cit. on p. 12).

[NAJ07] G. Noyel, J. Angulo, and D. Jeulin. “On distances, paths and connections for hyperspectral image segmentation”. In: *Proc. 8th Int. Symp. Math. Morphology (ISMM) 2007*. Ed. by G. Banon, J. Barrera, U. Braga-Neto, and N. Hirata. Vol. 1. São José dos Campos: Instituto Nacional de Pesquisas Espaciais (INPE), Oct. 2007, pp. 399–410. URL: http://urlib.net/dpi.inpe.br/ismm@80/2007/03.05.11.23 (cit. on p. 2).

[OS11] G. K. Ouzounis and P. Soille. “Pattern spectra from partition pyramids and hierarchies”. In: *Proc. 10th Int. Symp. Math. Morphology (ISMM) 2011*. Vol. 6671. Lecture Notes in Computer Science. 2011, pp. 108–119. DOI: 10.1007/978-3-642-21569-8_10 (cit. on pp. 8, 14, 17, 23, 39).

[OPS12] G.K. Ouzounis, M. Pesaresi, and P. Soille. “Differential area profiles: decomposition properties and efficient computation”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.8 (Aug. 2012), pp. 1533–1548. DOI: 10.1109/TPAMI.2011.245 (cit. on p. 1).

[OS10] G.K. Ouzounis and P. Soille. “Differential area profiles”. In: *20th Int. Conf. on Pattern Recognition, ICPR 2010*. Istanbul, Turkey, Aug. 2010, pp. 4085–4088. DOI: 10.1109/ICPR.2010.993 (cit. on p. 1).

[Ouz+12] G.K. Ouzounis, V. Syrris, L. Gueguen, and P. Soille. “The switchboard platform for interactive image information mining”. In: *Proc. of 8th Conference on Image Information Mining*. European Commission, Joint Research Centre, Oct. 2012, pp. 26–30. DOI: 10.2788/49465 (cit. on p. 39).

[OW07] G.K. Ouzounis and M.H.F. Wilkinson. “Mask-based second-generation connectivity and attribute filters”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.6 (2007), pp. 990–1004. DOI: 10.1109/TPAMI.2007.1045 (cit. on pp. 6, 19).

[OW10] G.K. Ouzounis and M.H.F. Wilkinson. “Partition-induced connections and operators for pattern analysis”. *Pattern Recognition* 43.10 (2010), pp. 3193–3207. DOI: 10.1016/j.patcog.200910.002 (cit. on pp. 1, 6, 13).

[PP93] N. Pal and S. Pal. “A review on image segmentation techniques”. *Pattern Recognition* 26.9 (1993), pp. 1277–1294. DOI: 10.1016/0031-3203(93)90135-J (cit. on p. 1).

[Pas+11] N. Passat, B. Naegel, F. Rousseau, M. Koob, and J.-L. Dietemann. "Semi-supervised learning for visual content analysis and understanding". *Pattern Recognition* 44.10-11 (2011), pp. 2539–2554. DOI: 10.1016/j.patcog.2011.03.025 (cit. on p. 1).

[PD05] E. Pekalska and R. Duin. *The Dissimilarity Representation for Pattern Recognition*. World Scientific, 2005 (cit. on p. 7).

[PS12] D. Poli and P. Soille. "Digital surface models extraction from ISPRS benchmark stereo data and refinement through constrained connectivity partitioning". *Photogrammetrie-Fernerkundung-Geoinformation* 2012.4 (Aug. 2012), pp. 317–329. DOI: 10.1127/1432-8364/2012/0120 (cit. on p. 36).

[Ron98] C. Ronse. "Set-theoretical algebraic approaches to connectivity in continuous digital spaces". *Journal of Mathematical Imaging and Vision* 8.1 (1998), pp. 41–58. DOI: 10.1007/s10851-008-0090-5 (cit. on p. 6).

[Ron08] C. Ronse. "Partial partitions, partial connections and connective segmentation". *Journal of Mathematical Imaging and Vision* 32.2 (Oct. 2008), pp. 97–125. DOI: 10.1007/s10851-008-0090-5 (cit. on p. 14).

[Ron11a] C. Ronse. "Idempotent block splitting on partial partitions, I: Isotone operators". *Order* 28.2 (July 2011), pp. 273–306. DOI: 10.1007/s11083-010-9171-3 (cit. on p. 14).

[Ron11b] C. Ronse. "Idempotent block splitting on partial partitions, II: Non-isotone operators". *Order* 28.2 (July 2011), pp. 307–339. DOI: 10.1007/s11083-010-9190-0 (cit. on p. 14).

[SG00] P. Salembier and L. Garrido. "Binary partition tree as an efficient representation for image processing, segmentation, and information retrieval". *IEEE Transactions on Image Processing* 9.4 (Apr. 2000), pp. 561–576. DOI: 10.1109/83.841934 (cit. on p. 2).

[SOG98] P. Salembier, A. Oliveras, and L. Garrido. "Anti-extensive connected operators for image and sequence processing". *IEEE Transactions on Image Processing* 7.4 (1998), pp. 555–570. DOI: 10.1109/83.663500 (cit. on pp. 1, 3, 12, 17).

[SS95] P. Salembier and J. Serra. "Flat zones filtering, connected operators, and filters by reconstruction". *IEEE Transactions on Image Processing* 4.8 (1995), pp. 1153–1160. DOI: 10.1109/83.403422 (cit. on pp. 1, 8, 18).

[SFP10] A. Schneider, M. Friedl, and D. Potere. "Monitoring urban areas globally using MODIS 500m data: new methods based on urban ecoregions". *Remote Sensing of Environment* 114.8 (Aug. 2010), pp. 1733–1746. DOI: 10.1016/j.rse.2010.03.003 (cit. on p. 35).

[Sch95] R. Schoenmakers. *Integrated Methodology for Segmentation of Large Optical Satellite Images in Land Applications of Remote Sensing*. Vol. EUR 16292 EN. European Commission, Joint Research Centre, Sept. 1995 (cit. on p. 2).

[Sed98] R. Sedgewich. *Algorithms in C*. Addison Wesley, 1998, pp. 90–109 (cit. on p. 18).

[Ser88] J. Serra, ed. *Image Analysis and Mathematical Morphology. II: Theoretical Advances*. London: Academic Press, 1988 (cit. on p. 5).

[Ser06] J. Serra. "A lattice approach to image segmentation". *Journal of Mathematical Imaging and Vision* 24.1 (Jan. 2006), pp. 83–130. DOI: 10.1007/s10851-005-3616-0 (cit. on pp. 6, 14).

[Sha98] C. Shaffer. *A practical introduction to data structures and algorithm analysis*. New Jersey: Prentice Hall, 1998, pp. 77–102 (cit. on p. 18).

[Soi07] P. Soille. "On genuine connectivity relations based on logical predicates". In: *Proc. of 14th Int. Conf. on Image Analysis and Processing, Modena, Italy*. IEEE Computer Society Press, Sept. 2007, pp. 487–492. DOI: 10.1109/ICIAP.2007.4362825 (cit. on pp. 3, 14, 15).

[Soi08] P. Soille. "Constrained connectivity for hierarchical image partitioning and simplification". *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.7 (July 2008), pp. 1132–1145. DOI: 10.1109/TPAMI.2007.70817 (cit. on pp. 2, 3, 8, 10, 14, 29, 36, 43).

[Soi09] P. Soille. "Recent developments in morphological image processing for remote sensing". In: *Image and Signal Processing for Remote Sensing*. Ed. by L. Bruzzone, J. Benediktsson, and S. Serpico. Vol. SPIE-7477. SPIE, 2009. DOI: 10.1117/12.836024 (cit. on p. 12).

[Soi10] P. Soille. "Constrained connectivity for the processing of very high resolution satellite images". *International Journal of Remote Sensing* 31.22 (July 2010). Erratum in the equation of page 5886: erosion (resp. dilation) must be replaced by gradient by erosion (resp. dilation)., pp. 5879–5893. DOI: 10.1080/01431161.2010.512622 (cit. on p. 36).

[Soi11] P. Soille. "Preventing chaining through transitions while favouring it within homogeneous regions". *Lecture Notes in Computer Science* 6671 (2011), pp. 96–107. DOI: 10.1007/978-3-642-21569-8_9 (cit. on pp. 2, 7, 14, 36).

[SG08] P. Soille and J. Grazzini. "Advances in constrained connectivity". *Lecture Notes in Computer Science* 4992 (Apr. 2008), pp. 423–433. DOI: 10.1007/978-3-540-79126-3_38 (cit. on p. 14).

[SG09] P. Soille and J. Grazzini. "Constrained connectivity and transition regions". *Lecture Notes in Computer Science* 5720 (2009), pp. 59–69. DOI: 10.1007/978-3-642-03613-2_6 (cit. on pp. 14, 15).

[SN12] P. Soille and L. Najman. "On morphological hierarchical representations for image processing and spatial data clustering". *Lecture Notes in Computer Science* 7346 (2012), pp. 43–67. DOI: 10.1007/978-3-642-32313-3_4 (cit. on p. 12).

[Tar75] R. Tarjan. "Efficiency of a good but not linear set union algorithm". *Journal of the ACM* 22 (1975), pp. 215–225. DOI: 10.1145/321879.321884 (cit. on pp. 3, 17, 18).

[Tar79] R. Tarjan. “Applications of path compression on balanced trees”. *Journal of the ACM* 26.4 (1979), pp. 690–715. DOI: 10.1145/322154.322161 (cit. on p. 3).

[Tar83] R. Tarjan. *Data Structures and Network Algorithms*. SIAM, 1983 (cit. on pp. 17, 18).

[Tho07] M. Thorup. “Equivalence between priority queues and sorting”. *Journal of the ACM* 54.6 (2007), 28:1–28:27. DOI: 10.1145/1314690.1314692 (cit. on p. 18).

[URW07] E.R. Urbach, J.B.T.M. Roerdink, and M.H.F. Wilkinson. “Connected shape-size pattern spectra for rotation and scale-invariant classification of gray-scale images”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.2 (2007), pp. 272–285. DOI: 10.1109/TPAMI.2007.28 (cit. on pp. 1, 19, 36).

[VMS08] V. Vilaplana, F. Marques, and Ph. Salembier. “Binary partition trees for object detection”. *IEEE Transactions on Image Processing* 17.11 (Nov. 2008), pp. 2201–2216. DOI: 10.1109/TIP.2008.2002841 (cit. on p. 2).

[WRW07] M.A. Westenberg, J.B.T.M. Roerdink, and M.H.F. Wilkinson. “Volumetric attribute filtering and interactive visualization using the max-tree representation”. *IEEE Transactions on Image Processing* 16.12 (2007), pp. 2943–2952. DOI: 10.1109/TIP.2007.909317 (cit. on p. 19).

[WO10] M.H.F. Wilkinson and G.K. Ouzounis. “Advances in connectivity and connected attribute filters”. In: *Advances in Imaging and Electron Physics*. Ed. by P. Hawkes. Vol. 163. Elsevier, Aug. 2010, pp. 219–222. DOI: 10.1016/S1076-5670(10)63010-8 (cit. on p. 6).

[WR00] M.H.F. Wilkinson and J.B.T.M. Roerdink. “Fast morphological attribute operations using Tarjan s union-find algorithm”. In: *Mathematical Morphology and its Applications to Image and Signal Processing*. Palo Alto, CA, June 2000, pp. 311–320 (cit. on pp. 17, 18).

[WW01] M.H.F. Wilkinson and M.A. Westenberg. “Shape preserving filament enhancement filtering”. In: *Proc. MICCAI’2001*. Ed. by W.J. Niessen and M.A. Viergever. Vol. 2208. Lecture Notes in Computer Science. 2001, pp. 770–777. DOI: 10.1007/3-540-45468-3_92 (cit. on pp. 19, 20).

[ZR03] J. Zunic and P. Rosin. “Rectilinearity measurements for polygons”. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.9 (2003), pp. 1193–1200. DOI: 10.1109/TPAMI.2003.1227997 (cit. on p. 36).

European Commission
EUR 25500 EN – Joint Research Centre – Institute for the Protection and Security of the Citizen

Title: The Alpha-Tree Algorithm

Authors: Georgios K. Ouzounis and Pierre Soille

Luxembourg: Publications Office of the European Union

2012 – 62 pp. – 21.0 x 29.7 cm

EUR – Scientific and Technical Research series – ISSN 1831-9424 (online), ISSN 1018-5593 (print)

ISBN 978-92-79-26279-1

doi:10.2788/48773

Abstract

A new multi-scale graph-space connectivity framework is presented. It is based upon a measure of dissimilarity between adjacent elements of the graph that is used to construct a hierarchy of partitions. Connected components or partition cells are either preserved or rejected based on a set of attribute criteria that are enumerated through logical predicates. Enforcing attribute constraints generates a dichotomy of the partition hierarchy that can be used for image segmentation and other related applications. The framework is supported by an efficient union-find based algorithm that delivers a tree representation of the totally ordered set of graph-space partitions. It is referred to as the Alpha-Tree. The practical complexity of the algorithm is linear with respect to the number of pixels. Processes on the tree can be launched interactively and in real-time, from a separate module detached from the tree construction phase. The type of attributes and attribute thresholds can be set and adjusted interactively. Timed experiments on highly complicated and massive satellite image tiles are presented, complemented by comparisons to the standard method.

LB-NA-25500-EN-N

As the Commission's in-house science service, the Joint Research Centre's mission is to provide EU policies with independent, evidence-based scientific and technical support throughout the whole policy cycle.

Working in close cooperation with policy Directorates-General, the JRC addresses key societal challenges while stimulating innovation through developing new standards, methods and tools, and sharing and transferring its know-how to the Member States and international community.

Key policy areas include: environment and climate change; energy and transport; agriculture and food security; health and consumer protection; information society and digital agenda; safety and security including nuclear; all supported through a cross-cutting and multi-disciplinary approach.

Publications Office

ISBN 978-92-79-26279-1